Laporan Tugas Kecil 3
IF2211 Strategi Algoritma
Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*
Semester II Tahun 2021/2022

Disusun oleh:

Jevant Jedidia Augustine
13520133

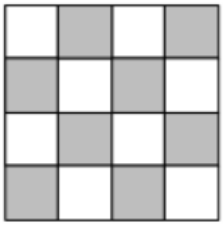
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022

Cara Kerja Program *Branch and Bound*

Sebelum dilakukan pencarian solusi dari 15-puzzle, perlu diperiksa terlebih dahulu apakah *initial state* dari puzzle yang dimasukkan dapat ditemukan solusinya. Untuk menentukan apakah puzzle dapat diselesaikan atau tidak, digunakan rumus berikut:

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

dimana,
KURANG(i) : banyaknya ubin bernomor *j* sedemikian sehingga *j* < *i* dan POSISI(*j*) > POSISI(*i*) dimana POSISI(*i*) adalah posisi dari ubin *i* pada puzzle yang diperiksa.
X : bernilai 1 apabila ubin yang kosong berada di daerah yang diarsir (pada gambar dibawah) dan 0 bila tidak.



Apabila nilai yang didapat dari rumus tersebut adalah ganjil, maka puzzle tidak bisa diselesaikan, bila nilai tersebut adalah genap, maka puzzle dapat diselesaikan

Apabila puzzle dapat diselesaikan, maka akan diterapkan algoritma *branch and bound* kepada puzzle tersebut. Akan dibangkitkan simpul-simpul berdasarkan kondisi dari puzzle yang akan dijadikan *parent* dari simpul-simpul tersebut. Simpul-simpul yang dibangun adalah puzzle dengan pemindahan ubin kosong ke arah tertentu. Pembangkitan simpul akan dilakukan dengan urutan pemindahan ubin kosong ke atas, kanan, bawah, dan kiri. Untuk setiap simpul yang dibangun, akan dilakukan perhitungan estimasi *cost* dari simpul tersebut dengan rumus berikut:

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

dimana,
f(P) : panjang lintasan dari simpul ke akar
g(P) : jumlah ubin tidak kosong yang tidak terdapat pada susunan goal puzzle

Simpul-simpul yang dibangun akan diperiksa terlebih dahulu apakah puzzle yang dibangun pada simpul tersebut sudah pernah dibangun atau belum, apabila sudah pernah maka simpul akan dimatikan. Simpul-simpul hidup yang dibangun akan dimasukkan ke *priority queue* dengan prioritas cost dari simpul tersebut (cost kecil akan didahulukan) kemudian nomor dari simpul tersebut (nomor kecil akan didahulukan). Setelah semua simpul yang dapat dibangun berdasarkan arah pergeseran ubin kosong telah dibangun, maka *priority queue* akan di-*pop head*-nya dan simpul yang di-*pop* akan dilakukan algoritma yang sama seperti algoritma

yang diterapkan terhadap simpul *initial state* dari puzzle. Algoritma *branch and bound* akan selesai dilakukan apabila sudah ditemukan state puzzle yang sama dengan goal.

Kode Program

puzzle.py

Fungsi readFile

```
def readFile(fileName): #Membaca file dan mengubahnya menjadi matrix 15-puzzle
    puzzle = []
    with open(fileName, 'r') as f:
        puzzle = [[int(angka) for angka in line.split(' ')] for line in f]
    return puzzle
```

Fungsi printPuzzle

```
def printPuzzle(puzzle): #Mencetak 15-puzzle
    for i in range (4):
        for j in range (4):
            if (puzzle[i][j] == 16):
                if j < 3:
                    print(' -', end = " ")
                else:
                    print(' -', end = "\n")
            else:
                if j < 3:
                    if puzzle[i][j] < 10:
                        print(" " + str(puzzle[i][j]), end = " ")
                    else:
                        print(str(puzzle[i][j]), end = " ")
                else:
                    if puzzle[i][j] < 10:
                        print(" " + str(puzzle[i][j]), end = "\n")
                    else:
                        print(str(puzzle[i][j]), end = "\n")
```

Fungsi convertToMatIndex

```
def convertToMatIndex(index): #Mengubah angka menjadi indeks pada matriks
    row = index // 4
    col = index % 4
    return row, col
```

Fungsi getKurang

```
def getKurang(puzzle): #Mendapatkan nilai kurang(i) dan index dari bagian yang kosong
    kurang = []
    for i in range (16):
        row, col = convertToMatIndex(i)
        ubin = puzzle[row][col]
        counter = 0
        if ubin == 16:
            indexKosong = i
        for j in range (i+1,16):
            row1, col1 = convertToMatIndex(j)
            temp = puzzle[row1][col1]
            if temp != 16:
                if ubin > temp:
                    counter += 1
        kurang.append([ubin,counter])
    return kurang, indexKosong
```

Fungsi getXKosong

```
def getXKosong(index): # Mendapatkan nilai X yang didapat dari bagian yang kosong
    arsir = [1,3,4,6,9,11,12,14]
    if index in arsir:
        return 1
    else:
        return 0
```

Fungsi countSolvable

```
def countSolvable(kurang, indexKosong): #Menghitung total dari kurang(i) ditambah dengan nilai X
    X = getXKosong(indexKosong)
    for i in range (16):
        X += kurang[i][1]
    return X
```

Fungsi sortKurang

```
def sortKurang(kurang): # Sort tabel kurang(i) sehingga i terurut menaik
    for i in range(16):
        if kurang[i][0] != i+1:
            for j in range (i,16):
                if kurang[j][0] == i+1:
                    temp = kurang[i]
                    kurang[i] = kurang[j]
                    kurang[j] = temp
                    break
```

Fungsi printKurang

```
def printKurang(kurang): #Mencetak kurang(i) dalam bentuk tabel
    print ("Ubin | Kurang(i)")
    print ("----------------")
    for i in range (16):
        if kurang[i][0] < 10:
            print (" " + str(kurang[i][0]) + "    |  " + str(kurang[i][1]))
        else:
            print (str(kurang[i][0]) + "    |  " + str(kurang[i][1]))
```

Fungsi indexKosong

```
def indexKosong(puzzle): #Mencari indeks dari ubin yang kosong
    for i in range(16):
        row, col = convertToMatIndex(i)
        if puzzle[row][col] == 16:
            return row, col
```

Fungsi checkMoveValid

```
def checkMoveValid(puzzle, direction): #Memeriksa apakah gerakan ubin kosong pada puzzle valid
    row, col = indexKosong(puzzle)
    if direction == "up":
        if row == 0:
            return False
        else:
            return True
    elif direction == "right":
        if col == 3:
            return False
        else:
            return True
    elif direction == "down":
        if row == 3:
            return False
        else:
            return True
    elif direction == "left":
        if col == 0:
            return False
        else:
            return True
```

Fungsi getCost

```
def getCost(puzzle): #Mendapatkan cost dari anak 15-puzzle
    counter = 0
    for i in range (16):
        row, col = convertToMatIndex(i)
        if puzzle[row][col] != 16:
            if puzzle[row][col] != i+1:
                counter += 1
    return counter
```

Fungsi movePuzzle

```
def movePuzzle(puzzleOG, direction): #Menggerakan ubin kosong pada puzzle sesuai dengan arahnya
    puzzle = [row[:] for row in puzzleOG]
    row, col = indexKosong(puzzle)
    if direction == "up":
        temp = puzzle[row-1][col]
        puzzle[row-1][col] = puzzle[row][col]
        puzzle[row][col] = temp
    elif direction == "right":
        temp = puzzle[row][col+1]
        puzzle[row][col+1] = puzzle[row][col]
        puzzle[row][col] = temp
    elif direction == "down":
        temp = puzzle[row+1][col]
        puzzle[row+1][col] = puzzle[row][col]
        puzzle[row][col] = temp
    elif direction == "left":
        temp = puzzle[row][col-1]
        puzzle[row][col-1] = puzzle[row][col]
        puzzle[row][col] = temp
    return puzzle
```

Fungsi checkGoal

```
def checkGoal(node): #Memeriksa apakah puzzle sudah sesuai dengan goal
    if node.cost - node.distance == 0:
        return True
    else:
        return False
```

Fungsi addSimpul

```
def addSimpul(simpulHidup, node): #Menambahkan simpul hidup dengan basis prioQueue
    if len(simpulHidup) != 0:
        index = len(simpulHidup)
        for i in range(len(simpulHidup)):
            if simpulHidup[i].cost > node.cost:
                index = i
                break
        simpulHidup.insert(index,node)
    else:
        simpulHidup.append(node)
```

Fungsi printSolution

```
def printSolution(root, goal): #mencetak path dari solusi 15-puzzle
    print("Initial:")
    printPuzzle(root.puzzle)
    print("\n")
    currentNode = root
    for route in goal.path:
        for child in currentNode.children:
            if child.name == route:
                print(child.name)
                printPuzzle(child.puzzle)
                print("\n")
                currentNode = child
                break
```

Fungsi checkAccessed

```
def checkAccessed(accessed, puzzle): #Periksa apakah suatu state dari puzzle sudah pernah dibuat
    if puzzle in accessed:
        return True
    return False
```

Fungsi findSolution

```
def findSolution(puzzle): #mencari solusi dari 15-puzzle
    simpulHidup = []
    accessed = []
    found = False
    gerakan = ["up", "right", "down", "left"]
    root = TreeNode("root", puzzle, 0, 0, [])
    simpulHidup.append(root)
    accessed.append(root.puzzle)
    if checkGoal(root):
        found = True
        solution = root

    while (len(simpulHidup) != 0 and not found):
        currentNode = simpulHidup.pop(0)
        newDis = currentNode.distance+1
        for item in gerakan:
            newPath = currentNode.path.copy()
            if checkMoveValid(currentNode.puzzle, item):
                newPuzzle = movePuzzle(currentNode.puzzle,item)
                if not checkAccessed(accessed, newPuzzle):
                    newPath.append(item)
                    accessed.append(newPuzzle)
                    child = TreeNode(item, newPuzzle, newDis, getCost(newPuzzle), newPath)
                    currentNode.addChild(child)
                    addSimpul(simpulHidup,child)

                    if checkGoal(child):
                        found = True
                        solution = child
                        break

    printSolution(root, solution)
```

pohon.py

```
class TreeNode:
    id = 0
    def __init__(self, name, puzzle, distance, cost, path):
        self.name = name
        TreeNode.id += 1
        self.id = TreeNode.id
        self.children = []
        self.puzzle = puzzle
        self.distance = distance
        self.cost = cost + distance
        self.path = path

    def addChild(self, obj):
        self.children.append(obj)
```

main.py

```
from puzzle import *
from pohon import *
import time
import os

fileName = input("Masukkan nama file: ")
cwd = os.getcwd()
print(cwd)
puzzle = readFile(".\\testcase\\" + fileName)

# Nomor 1
print("Matrix awal: ")
printPuzzle(puzzle)
print("\n")

# Nomor 2
start = time.time()
kurang, indexKosong = getKurang(puzzle)
sortKurang(kurang)
end = time.time()
totalTime = end-start
printKurang(kurang)

# Nomor 3
print("Nilai dari KURANG(i) + X:", end = " ")
start = time.time()
X = countSolvable(kurang, indexKosong)
end = time.time()
totalTime += end-start
print(X)
```

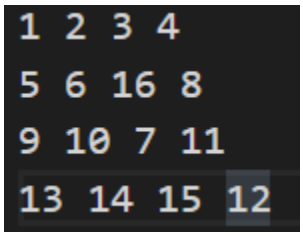
```
if X % 2 == 0:
    # Nomor 5
    start = time.time()
    findSolution(puzzle)
    end = time.time()
    totalTime += end-start
else:
    # Nomor 4
    print("Tidak dapat diselesaikan")

# Nomor 6
print("Waktu eksekusi program: " + str(totalTime) + " s")

# Nomor 7
if X % 2 == 0:
    print("Jumlah simpul yang dibangkitkan: " + str(TreeNode.id-1))
    print("\n")
else:
    print("Jumlah simpul yang dibangkitkan: 0" )
    print("\n")
```

Input-Output Program

Untuk masukan, ubin bernilai 16 merupakan ubin yang kosong.

solvable1.txt

Input:



Output:

```
Masukkan nama file: solvable1.txt
Matrix awal:
 1  2  3  4
 5  6  -  8
 9 10  7 11
13 14 15 12


Ubin | Kurang(i)
---------------
 1   |  0
 2   |  0
 3   |  0
 4   |  0
 5   |  0
 6   |  0
 7   |  0
 8   |  1
 9   |  1
10   |  1
11   |  0
12   |  0
13   |  1
14   |  1
15   |  1
16   |  9
Nilai dari KURANG(i) + X: 16


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

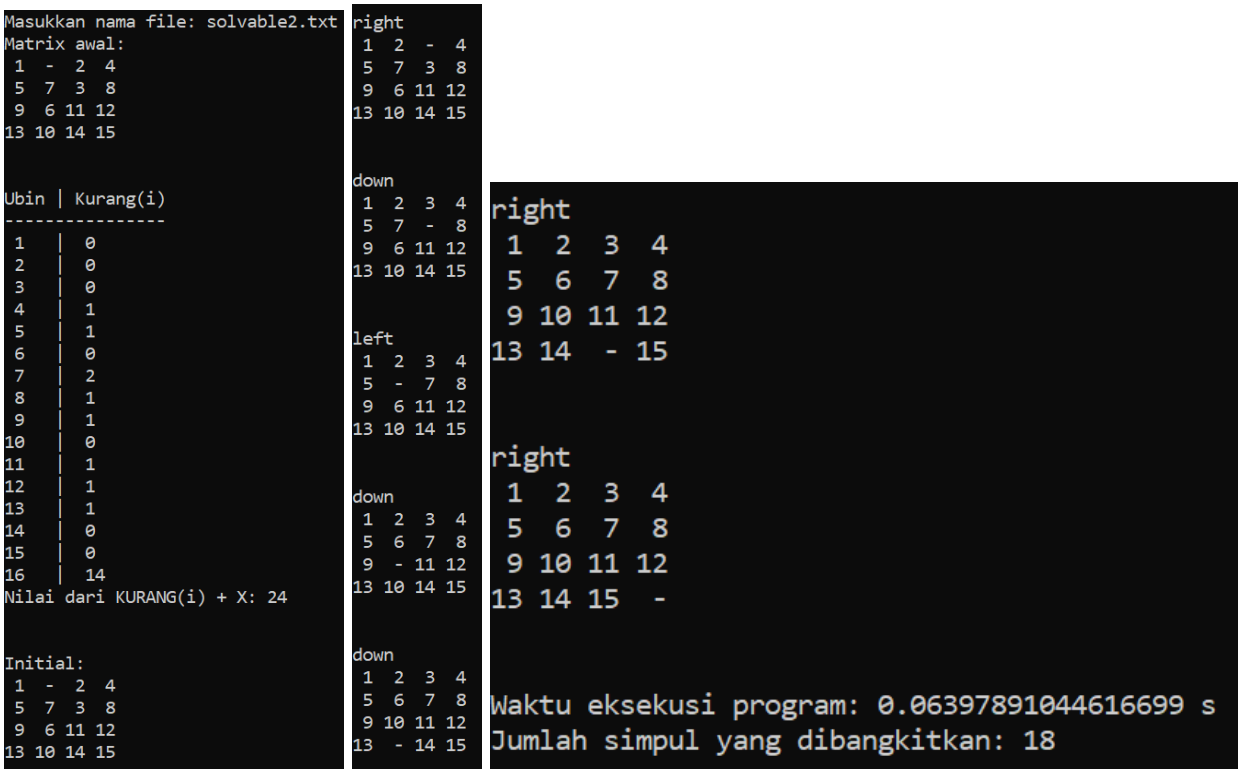
```
down
 1  2  3  4
 5  6  7  8
 9 10  - 11
13 14 15 12


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


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


Waktu eksekusi program: 0.028191089630126953 s
Jumlah simpul yang dibangkitkan: 9
```

solvable2.txt

Input:

```
1 16 2 4
5 7 3 8
9 6 11 12
13 10 14 15
```

Output:



solvable3.txt

Input:

```
1 2 3 4
5 6 16 7
9 10 8 11
13 14 12 15
```

Output:

```
Masukkan nama file: solvable3.txt
Matrix awal:
 1  2  3  4
 5  6  -  7
 9 10  8 11
13 14 12 15


Ubin | Kurang(i)
---------------
 1   |  0
 2   |  0
 3   |  0
 4   |  0
 5   |  0
 6   |  0
 7   |  0
 8   |  0
 9   |  1
10   |  1
11   |  0
12   |  0
13   |  1
14   |  1
15   |  0
16   |  9
Nilai dari KURANG(i) + X: 14


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

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


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


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


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


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

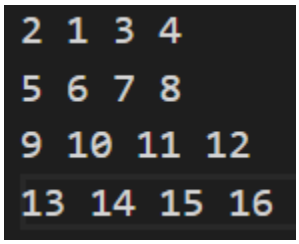
```
up
 1  2  3  4
 5  6  7 11
 9  - 12  8
13 10 14 15


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


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


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


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

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


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


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


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


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

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


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


Waktu eksekusi program: 1.8561983108520508 s
Jumlah simpul yang dibangkitkan: 6091
```

notsolvable1.txt

Input:

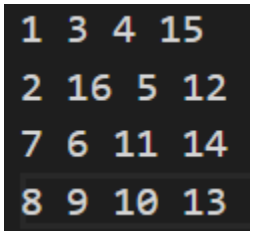


Output:

```
Masukkan nama file: notsolvable1.txt
Matrix awal:
 2  1  3  4
 5  6  7  8
 9 10 11 12
13 14 15  -


Ubin | Kurang(i)
---------------
 1   |  0
 2   |  1
 3   |  0
 4   |  0
 5   |  0
 6   |  0
 7   |  0
 8   |  0
 9   |  0
10   |  0
11   |  0
12   |  0
13   |  0
14   |  0
15   |  0
16   |  0
Nilai dari KURANG(i) + X: 1


Tidak dapat diselesaikan
Waktu eksekusi program: 0.0009961128234863281 s
Jumlah simpul yang dibangkitkan: 0
```

notsolvable2.txt

Input:



Output:

```
Masukkan nama file: notsolvable2.txt
Matrix awal:
 1  3  4 15
 2  -  5 12
 7  6 11 14
 8  9 10 13


Ubin | 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
Nilai dari KURANG(i) + X: 37


Tidak dapat diselesaikan
Waktu eksekusi program: 0.0 s
Jumlah simpul yang dibangkitkan: 0
```

Link github: https://github.com/JevantJedidia/Tucil3_13520133

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		✓