Laporan

Tugas Kecil 3 IF2211 Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Disusun Oleh:

Aldwin Hardi Swastia - 13520167

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

ALGORITMA BRANCH AND BOUND

1. Cek apakah puzzle dapat diselesaikan dengan rumus $\sum_{n-1}^{16} KURANG(i) + X$ dimana nilai X adalah 1 jika sel kosong berada pada ubin ganjil dan 0 jika sel kosong berada pada ubin genap. Fungsi $KURANG(i)$ adalah banyaknya ubin bernomor j sedemikian sehingga $j < i$ dan $POSISI(j) > POSISI(i)$. Dimana fungsi $POSISI(i)$ merupakan posisi ubin bernomor i pada susunan yang diperiksa.
2. Puzzle hanya dapat diselesaikan apabila nilai $\sum_{n-1}^{16} KURANG(i) + X$ genap.
3. Selanjutnya apabila puzzle dapat diselesaikan, *cost* setiap simpul dihitung dengan $\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$ dimana $\hat{f}(i)$ merupakan kedalaman pohon dan $\hat{g}(i)$ merupakan jumlah ubin yang tidak terdapat pada susunan akhir.
4. Pertama, masukan state simpul utama pada queue (pada kasus ini digunakan priority queue untuk memilih cost yang terendah).
5. Kemudian dequeue elemen dari queue dan cek apabila state simpul merupakan goal. Jika bukan merupakan goal masukan state simpul pada visited. Sedangkan jika merupakan simpul goal stop.
6. Cek semua kemungkinan pergerakan dari puzzle tersebut, kemudian cek setiap pergerakan pada visited.
7. Apabila pergerakan belum ada pada dalam visited, bangkitkan simpul tersebut dan masukan ke dalam queue.
8. Lakukan langkah ke-4 hingga ditemukan goal node.
9. Jika sudah ditemukan goal node, program selesai.

SOURCE CODE

Bahasa Pemrograman: Python

main.py

```python
from numpy import matrix
from inputPuzzle import randomInput, inputFile
from puzzleSolver import *
import time

print("""
--- 15-Puzzle Solver ---
1. Random Input
2. File Input
3. Exit
""")

# Input Puzzle
inputOptions = input("Pilih: ")
while(inputOptions != "1" and inputOptions != "2" and inputOptions != "3"):
    print("Pilihan tidak tersedia")
    inputOptions = input("Pilih: ")
if(inputOptions == "1"):
    matrix = randomInput()
elif(inputOptions == "2"):
    matrix = inputFile()
elif(inputOptions == "3"):
    exit()

print("Matrix :")
printMatrix(matrix)



# Print all Kurang(i)
initialKurang(matrix)
print()

# Check if puzzle is solvable
if(isSolveable(matrix)):
    # Solve puzzle
    print("Puzzle dapat diselesaikan")
    print("Solving...\n")
    timeStart = time.time()
    solution, nodeCount = solve(matrix)
    timeEnd = time.time()

    # Initialize result
    result = []
```

```
    # Backtrack to root
    while(solution.parent != None):
        result.append(solution.move)
        solution = solution.parent

    # Reverse result
    result.reverse()

    # Print result
    printSteps(matrix, result)
    print("All Steps:", result)
    print("Total Steps:", len(result))
    print("Node Count:", nodeCount)

    print("Time: ", timeEnd-timeStart)
else:
    # Puzzle unsolvable
    print("Puzzle tidak dapat diselesaikan")
```

puzzleSolver.py

```
import copy
from solverLibrary import PrioQueue, StateNode
import numpy as np

def solve(matrix):
    # Branch and Bound Algorithm Solver

    # Initialize NodeCount as 1 (root)
    NodeCount = 1

    # Initialize Queue
    PQ = PrioQueue(lambda x,y: x.cost < y.cost)

    # Initialize solutionNode as None
    solutionNode = None

    # Initialize visited
    visited = {}

    # Initialize goalMatrix
    goalMatrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

    # Initialize root
    root = StateNode()
    root.matrix = matrix
    root.cost = calculateCost(matrix)
```

```
    # Add root to queue
    PQ.enqueue(root)

    # While queue is not empty
    while(not PQ.is_empty()):
        # Dequeue
        node = PQ.dequeue()

        # Check if node is goal
        if(node.matrix == goalMatrix):
            # Add node to result
            solutionNode = node
            break

        # Add node to visited
        visited[str(node.matrix)] = True

        # Get all possible moves
        moves = getMoves(node.matrix)

        # For each move
        for move in moves:

            # Move matrix
            moveMatrix = movePuzzle(node.matrix, move)

            # Check if moveMatrix is in visited
            if(str(moveMatrix) not in visited):
                # Create new node
                newNode = StateNode()
                newNode.matrix = moveMatrix
                newNode.parent = node
                newNode.depth = node.depth + 1
                newNode.cost = calculateCost(moveMatrix) + newNode.depth
                newNode.move = move

                # Add new node to queue
                PQ.enqueue(newNode)
                NodeCount += 1

    return solutionNode, NodeCount

def getMoves(matrix):
    # Get all possible moves from a state matrix
    moves = []
    row,col = find(16,matrix)
    if(row > 0):
```

```
        moves.append("up")
    if(row < 3):
        moves.append("down")
    if(col > 0):
        moves.append("left")
    if(col < 3):
        moves.append("right")

    return moves

def calculateCost(matrix):
    # Calculate cost of a state matrix
    cost = 0
    reshaped = np.reshape(matrix,(16,))
    for i in range(16):
        if(reshaped[i] != i+1):
            cost += 1
    return cost

def movePuzzle(matrix, move):
    # Move matrix according to move
    row,col = find(16,matrix)

    movedMatrix = copy.deepcopy(matrix)
    if(move == "up"):
        movedMatrix[row][col], movedMatrix[row-1][col] = movedMatrix[row-
1][col], movedMatrix[row][col]
    elif(move == "down"):
        movedMatrix[row][col], movedMatrix[row+1][col] =
movedMatrix[row+1][col], movedMatrix[row][col]
    elif(move == "left"):
        movedMatrix[row][col], movedMatrix[row][col-1] = movedMatrix[row][col-
1], movedMatrix[row][col]
    elif(move == "right"):
        movedMatrix[row][col], movedMatrix[row][col+1] =
movedMatrix[row][col+1], movedMatrix[row][col]

    return movedMatrix



def isSolveable(matrix):
    #Check if puzzle is solvable

    row,col = find(16,matrix)
    if((row+col)%2 == 0):
        X = 0
    else:
        X = 1
```

```
    KurangX = kurang(matrix) + X
    print("sum Kurang(i) + X: ", KurangX)

    if ((KurangX)%2 == 0):
        return True
    else:
        return False

def kurang(matrix):
    # Menghitung total kurang dari sebuah matrix
    total = 0
    reshaped = np.reshape(matrix,(16,))
    for i in range(16):
        for j in range(i,16):
            if(reshaped[i] > reshaped[j]):
                total += 1

    return total

def initialKurang(matrix):
    # Mencetak setiap kurang dari sebuah matrix
    reshaped = np.reshape(matrix,(16,))
    dict = {}
    for i in range(16):
        kurangi = 0
        for j in range(i,16):
            if(reshaped[i] > reshaped[j]):
                kurangi += 1
        dict[reshaped[i]] = kurangi

    for i in range(16):
        print("Kurang("+ str(i+1) +") = ", dict[i+1])


def find(x, matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if(matrix[i][j] == x):
                return i,j
    return -1,-1

def printMatrix(matrix):
    for i in range(4):
        for j in range(4):
            if(matrix[i][j] == 16):
                print("-", end=" ")
            else:
                print(matrix[i][j], end=" ")
```

```
        print()

def printSteps(matrix, result):
    # Print steps to solve the puzzle
    curMatrix = copy.deepcopy(matrix)
    print("Steps to solve the puzzle: \n")
    for i in range(len(result)):
        print("Step " + str(i+1) + ": " + result[i])
        curMatrix = movePuzzle(curMatrix, result[i])
        printMatrix(curMatrix)
        print()
```

inputPuzzle.py

```
def inputFile():
    # Masukan file
    print("Masukan nama file: ", end="")
    fileName = input()
    file = open("./tes/"+fileName, "r")
    matrix = []
    for line in file:
        matrix.append(list(line.split()))

    for i in range(4):
        for j in range(4):
            if(matrix[i][j] == '-' or matrix[i][j] == '0'):
                matrix[i][j] = 16
            else:
                matrix[i][j] = int(matrix[i][j])

    return matrix

def randomInput():
    # Masukan random
    import numpy as np
    matrix = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
    matrix = matrix.ravel()
    np.random.shuffle(matrix)
    matrix = matrix.reshape(4,4)
    matrix = matrix.tolist()

    return matrix
```

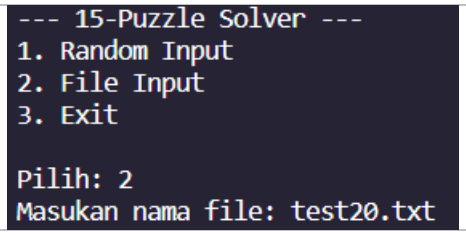
solverLibrary.py

```
class PrioQueue:
    # Constructor
```

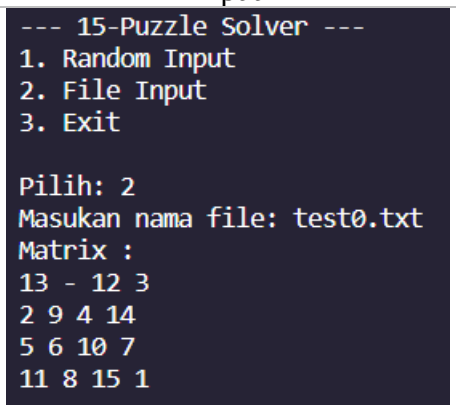
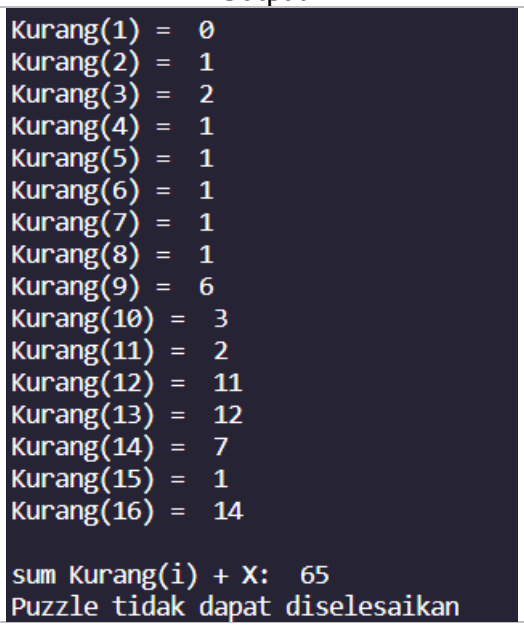
```
    def __init__(self, priority_function):
        self.queue = []
        self.func = priority_function

    # Check if queue is empty
    def is_empty(self):
        return len(self.queue) == 0

    # Enqueue item with priority
    def enqueue(self, item):
        i = 0
        found = False

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

        self.queue.insert(i, item)

    # Dequeue
    def dequeue(self):
        return self.queue.pop(0)

class StateNode:
    def __init__(self):
        self.matrix = []
        self.parent = None
        self.depth = 0
        self.cost = 0
        self.move = ""
```

SCREENSHOT INPUT OUTPUT

1. Main Menu

Main


2. test0.txt

Input	Output
	

3. testbri0.txt

Input	Output

```
--- 15-Puzzle Solver ---
1. Random Input
2. File Input
3. Exit

Pilih: 2
Masukan nama file: testbri0.txt
Matrix :
13 6 11 2
4 15 1 3
- 12 7 8
5 9 14 10
```

```
Kurang(1) =  0
Kurang(2) =  1
Kurang(3) =  0
Kurang(4) =  2
Kurang(5) =  0
Kurang(6) =  5
Kurang(7) =  1
Kurang(8) =  1
Kurang(9) =  0
Kurang(10) =  0
Kurang(11) =  9
Kurang(12) =  5
Kurang(13) =  12
Kurang(14) =  1
Kurang(15) =  9
Kurang(16) =  7

sum Kurang(i) + X:  53
Puzzle tidak dapat diselesaikan
```

4. test10.txt

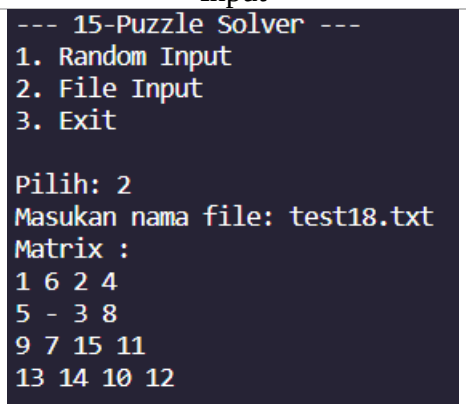
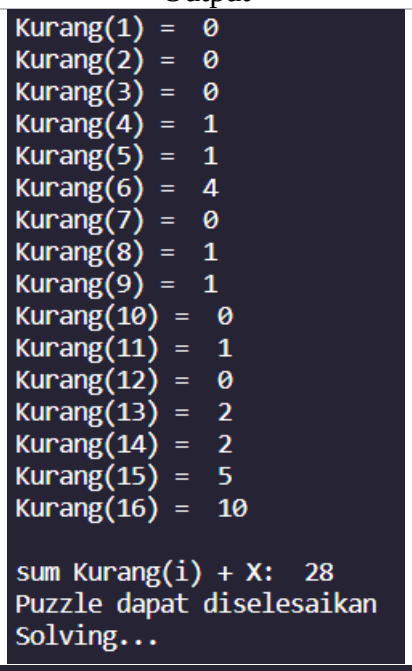
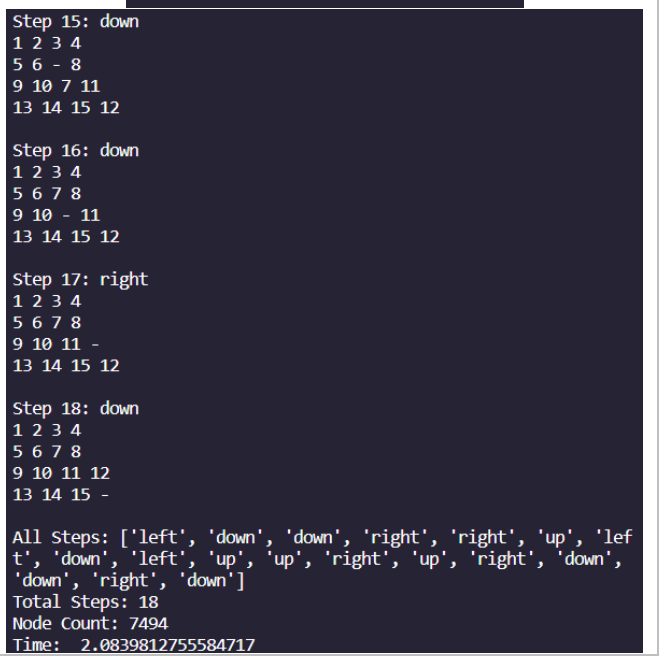
Input	Output

```
--- 15-Puzzle Solver ---
1. Random Input
2. File Input
3. Exit

Pilih: 2
Masukan nama file: test10.txt
Matrix :
5 1 3 4
9 2 7 8
- 6 15 11
13 10 14 12
```

```
Kurang(1) =  0
Kurang(2) =  0
Kurang(3) =  1
Kurang(4) =  1
Kurang(5) =  4
Kurang(6) =  0
Kurang(7) =  1
Kurang(8) =  1
Kurang(9) =  4
Kurang(10) =  0
Kurang(11) =  1
Kurang(12) =  0
Kurang(13) =  2
Kurang(14) =  1
Kurang(15) =  5
Kurang(16) =  7

sum Kurang(i) + X:  28
Puzzle dapat diselesaikan
Solving...
```

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

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

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

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

All Steps: ['up', 'up', 'right', 'down', 'down', 'down', 'right', 'up', 'right', 'down']
Total Steps: 10
Node Count: 25
Time:  0.0010001659393310547
```

5. test18.txt

Input	Output
	 

6. test21.txt

Input	Output

```
--- 15-Puzzle Solver ---
1. Random Input
2. File Input
3. Exit

Pilih: 2
Masukan nama file: test21.txt
Matrix :
2 3 4 11
1 5 10 8
9 6 12 15
13 14 - 7
```

```
Kurang(1) =  0
Kurang(2) =  1
Kurang(3) =  1
Kurang(4) =  1
Kurang(5) =  0
Kurang(6) =  0
Kurang(7) =  0
Kurang(8) =  2
Kurang(9) =  2
Kurang(10) =  4
Kurang(11) =  7
Kurang(12) =  1
Kurang(13) =  1
Kurang(14) =  1
Kurang(15) =  3
Kurang(16) =  1
```

```
Step 18: left
1 2 3 4
5 6 - 8
9 10 7 11
13 14 15 12

Step 19: down
1 2 3 4
5 6 7 8
9 10 - 11
13 14 15 12

Step 20: right
1 2 3 4
5 6 7 8
9 10 11 -
13 14 15 12

Step 21: down
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 -

All Steps: ['right', 'up', 'left', 'up', 'right', 'up', 'left',
'left', 'left', 'down', 'right', 'down', 'right', 'down', 'right
', 'up', 'up', 'left', 'down', 'right', 'down']
Total Steps: 21
Node Count: 11560
Time:  4.861548185348511
```

SOURCE CODE FILE

https://github.com/aldwinhs/Branch-and-Bound-15-Puzzle

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		√