

LAPORAN TUGAS KECIL 3

IF2211 STRATEGI ALGORITMA

“Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound”



NIM : 13520132

Nama : Januar Budi Ghifari

Kelas : K03

A. Algoritma Divide and Conquer

Algoritma Branch and Bound adalah algoritma yang biasa digunakan untuk persoalan optimasi. Optimasi yang dimaksud di sini adalah meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (constraints) persoalan tersebut.

Pada penerapannya, Algoritma Branch and Bound memiliki konsep yang mirip dengan BFS (Breadth First Search), dimana penerapannya dilakukan dengan konsep pohon dengan akar dan penyelesaiannya persoalan dilakukan dengan menelusuri simpul-simpul anaknya. Berbeda dengan algoritma BFS yang ekspansi simpul berikutnya didasarkan pada urutan pembangkitannya, algoritma Branch and Bound melakukan ekspansi simpul terhadap simpul dengan biaya atau cost terkecil. Penghitungan biaya pada simpul ini berbeda-beda menyesuaikan dengan persoalan yang ingin diselesaikan.

Program ini dibuat untuk menyelesaikan persoalan 15-Puzzle dengan algoritma Branch and Bound menggunakan Bahasa pemrograman Python. Berikut deskripsi singkat cara kerja program saya.

1. Program berjalan dengan memanggil solver() yang merupakan implementasi dari kelas BranchAndBound
2. Sebagai inisiasi, Matrix masukan akan dicek dulu syarat penyelesaian 15-puzzlenya dengan fungsi isSolvable yang akan mengeceknya dengan fungsi kurangI dan oneOrZero, apabila hasil genap, maka solvable.
3. Apabila solvable, program akan berlanjut dengan membangkitkan simpul-simpul anak berdasarkan fungsi move yang ada. Selanjutnya simpul simpul akan dibandingkan berdasarkan hasil fungsi cost terkecil.
4. Simpul dengan cost terkecil akan menjadi parent baru, dan simpul-simpul yang sudah dicek sebelumnya akan disimpan data list dengan konsep hashmap untuk menghemat waktu apabila ditemukan simpul yang sama.
5. Parent baru akan dimunculkan simpul-simpul anaknya yang kemudian akan dibandingkan lagi seperti step ke 4
6. Simpul-simpul baru akan terus dimunculkan sampai terbentuk simpul yang sama dengan finalState setelah itu algoritma berhenti dan dilanjutkan dengan print path dari posisi awal ke finalState.

B. Source Code Program

main

```
import numpy as np
from queue import PriorityQueue
import os.path
import time

class Node:
    def __init__(self, move, cost):
        self.move = move
        self.matrix = np.arange(16).reshape(4,4)
        self.cost = cost

    def readfile(self):
        numList = []
        while True:
            filename = input("Masukan nama file dengan ekstensinya (ex: test1.txt): ")
            path = "puzzle/" + filename
            if (os.path.isfile(path)):
                break
            else :
                print("File tidak ditemukan! silahkan input ulang")
        file = open(path)
        for i in range(4) :
            numList.extend([int(number) for number in
file.readline().split()])
        for i in range(len(numList)):
            if numList[i] == 16:
# Input blank boleh 0 atau 16, nanti akan diubah menjadi 0
                numList[i] = 0
        self.matrix = np.array(numList).reshape((4,4))

    def locateZero(self):
        result = np.where(self.matrix == 0)
        row = result[0][0]
        col = result[1][0]
        return row, col

    def printMatrix(self):
        print("-----")
        for arr in self.matrix :
            for angka in arr :
                print("|",end="")
                if angka == 0 :
                    print(" ",end=" ")
                elif angka < 10 :
                    print(" ",angka,end=" ")
```

```

        else :
            print("",angka,end=" ")
            print("|")
            print("-----")

def oneOrZero(self):
    row, col = self.locateZero()
    return (row + col)% 2

def kurangI(self):
    r, c = self.locateZero()
    tempMatrix = self.matrix.copy()
    tempMatrix[r][c] = 16
    tempMatrix = tempMatrix.flatten()
    total = 0
    arr = [0 for i in range(16)]
    for i in range (16):
        curNumber = tempMatrix[i]
        count = 0
        for j in range(i + 1, len(tempMatrix)):
            if tempMatrix[j] < curNumber:
                count += 1
        arr[tempMatrix[i] - 1] = count
    return arr

def syarat(self):
    total = sum(self.kurangI())
    return int(total + self.oneOrZero())

def isSolvable(self):
    if (self.syarat()%2 == 0):
        return True
    else:
        return False

def countSame(self):
    temp = self.matrix.flatten()
    count = 0
    for i in range(16) :
        if (temp[i] != (i+1) and temp[i] != 0):
            count += 1
    return count

#MOVEMENT

def moveUp(self):
    row, col = self.locateZero()
    after = self.matrix.copy()
    after[row-1,col] = self.matrix[row,col]
    after[row,col] = self.matrix[row-1,col]
    return after

def moveDown(self):
    row, col = self.locateZero()
    after = self.matrix.copy()
    after[row+1,col] = self.matrix[row,col]
    after[row,col] = self.matrix[row+1,col]
    return after

```

```

        return after

    def moveLeft(self):
        row, col = self.locateZero()
        after = self.matrix.copy()
        after[row,col-1] = self.matrix[row,col]
        after[row,col] = self.matrix[row,col-1]
        return after

    def moveRight(self):
        row, col = self.locateZero()
        after = self.matrix.copy()
        after[row,col+1] = self.matrix[row,col]
        after[row,col] = self.matrix[row,col+1]
        return after

    def __lt__(self, other):
        return False

class BranchAndBound:
    def __init__(self):
        self.checked = []
        self.queue = PriorityQueue()
        self.mapMatrix = {}
        self.root = Node(["-"], 0)
        self.finalState = Node(["-"], 0)
        self.target = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,0]])

    def bAndB(self):
        while True:
            curNode = self.queue.get()[1]
            self.checked.append(curNode)
            row, col = curNode.locateZero()
            curMove = curNode.move
            if np.array_equal(curNode.matrix, self.target):
                self.finalState = curNode
                break

            #MOVE RIGHT
            if(col != 3 and curNode.move[-1] != "left"):
                newNode = Node(curMove + ["right"], len(curNode.move))
                newNode.matrix = curNode.moveRight()
                newNode.cost += newNode.countSame()
                if newNode.matrix.tobytes() not in self.mapMatrix.keys():
                    self.mapMatrix[newNode.matrix.tobytes()] =
                    True
                    self.queue.put((newNode.cost, newNode))
                    if np.array_equal(newNode.matrix, self.target):
                        self.finalState = newNode
                        break

            #MOVE DOWN
            if(row != 3 and curNode.move[-1] != "up"):

                newNode = Node(curMove + ["down"], len(

```

```

newNode = Node(curMove + ["down"], len(
curNode.move))

newNode.matrix = curNode.moveDown()
newNode.cost += newNode.countSame()

if newNode.matrix.tobytes() not in self
.mapMatrix.keys():

    self.mapMatrix[newNode.matrix.tobytes()] =
True

    self.queue.put((newNode.cost, newNode))

    if np.array_equal(newNode.matrix, self
.target):

        self.finalState = newNode
        break

#MOVE LEFT

if(col != 0 and curNode.move[-1] != "right"):

    newNode = Node(curMove + ["left"], len(
curNode.move))

    newNode.matrix = curNode.moveLeft()
    newNode.cost += newNode.countSame()

    if newNode.matrix.tobytes() not in self
.mapMatrix.keys():

        self.mapMatrix[newNode.matrix.tobytes()] =
True

        self.queue.put((newNode.cost, newNode))

        if np.array_equal(newNode.matrix, self
.target):

            self.finalState = newNode
            break

#MOVE UP

if(row != 0 and curNode.move[-1] != "down"):

    newNode = Node(curMove + ["up"], len(
curNode.move))

    newNode.matrix = curNode.moveUp()
    newNode.cost += newNode.countSame()

    if newNode.matrix.tobytes() not in self
.mapMatrix.keys():

        self.mapMatrix[newNode.matrix.tobytes()] =
True

        self.queue.put((newNode.cost, newNode))

        if np.array_equal(newNode.matrix, self
.target):

            self.finalState = newNode
            break

def printPath(self):
    print("Banyaknya simpul yang dibangkitkan : ", self
.queue.qsize() + len(self.checked))
    print("Banyaknya step yang ditempuh : ", len(
self.finalState.move) - 1)
    print()
    temp = self.root
    for i in range(1, len(self.finalState.move)):
        if self.finalState.move[i] == "up":
            print("Up")
            temp.matrix = temp.moveUp()
        elif self.finalState.move[i] == "down":
            print("Down")
            temp.matrix = temp.moveDown()
        elif self.finalState.move[i] == "left":
            print("Left")
            temp.matrix = temp.moveLeft()
        elif self.finalState.move[i] == "right":
            print("Right")
            temp.matrix = temp.moveRight()

```

```
        elif self.finalState.move[i] == "down":
            print("Down")
            temp.matrix = temp.moveDown()
        elif self.finalState.move[i] == "left":
            print("Left")
            temp.matrix = temp.moveLeft()
        elif self.finalState.move[i] == "right":
            print("Right")
            temp.matrix = temp.moveRight()
        temp.printMatrix()
        print()

def solver(self):
    self.root.printMatrix()
    kurangi = self.root.kurangI()
    for i in range(16):
        print("Kurang(" + str(i+1)+ ")" + " = " + str(kurangi
[i]))
        print("syarat = ", str(self.root.syarat()))
        if (self.root.isSolvable()):
            self.root.cost = self.root.countSame()
            self.queue.put((self.root.cost, self.root))
            self.mapMatrix[self.root.matrix.tobytes()] = True
            self.bAndB()
        else:
            print("Puzzle tidak solvable")

puzzle = BranchAndBound()
puzzle.root.readFile()
waktuawal = time.time()
puzzle.solver()
waktuakhir = time.time()
if puzzle.root.isSolvable():
    puzzle.printPath()
    print("Waktu eksekusi : ", "%.4f" %(waktuakhir-waktuawal), "sekon")
```

C. Output

gabisa.txt

```
Kurang(2) = 0
Kurang(3) = 1
Kurang(4) = 1
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 1
Kurang(8) = 0
Kurang(9) = 0
Kurang(10) = 0
Kurang(11) = 3
Kurang(12) = 6
Kurang(13) = 0
Kurang(14) = 4
Kurang(15) = 11
Kurang(16) = 10
syarat = 37
Puzzle tidak solvable
Waktu eksekusi : 0.0070 sekon
```

input1.txt

Masukan nama file dengan ekstensinya (ex: test1.txt): input1.txt

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

```
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) = 0
Kurang(12) = 2
Kurang(13) = 0
Kurang(14) = 0
Kurang(15) = 0
Kurang(16) = 8
syarat = 12
Banyaknya simpul yang dibangkitkan : 1621
Banyaknya step yang ditempuh : 16
```

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 | |

Right

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

Waktu eksekusi : 0.0780 sekon

gabisa2.txt

```
Masukan nama file dengan ekstensinya (ex: test1.txt): gabisa2.txt
-----
|   | 3 | 15 | 14 |
-----
| 1 | 4 | 2 | 11 |
-----
| 7 | 5 | 12 | 6 |
-----
| 8 | 9 | 10 | 13 |
-----
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 2
Kurang(4) = 1
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 2
Kurang(8) = 0
Kurang(9) = 0
Kurang(10) = 0
Kurang(11) = 6
Kurang(12) = 4
Kurang(13) = 0
Kurang(14) = 12
Kurang(15) = 13
Kurang(16) = 15
syarat = 55
Puzzle tidak solvable
Waktu eksekusi : 0.0020 sekon
```

```
input2.txt
Masukan nama file dengan ekstensinya (ex: test1.txt): input2.txt
-----
| 2 | 5 | 3 | 4 |
-----
| 1 | 6 | 15 |   |
-----
| 9 | 10 | 8 | 7 |
-----
| 13 | 14 | 12 | 11 |
-----
Kurang(1) = 0
Kurang(2) = 1
Kurang(3) = 1
Kurang(4) = 1
Kurang(5) = 3
Kurang(6) = 0
Kurang(7) = 0
Kurang(8) = 1
Kurang(9) = 2
Kurang(10) = 2
Kurang(11) = 0
Kurang(12) = 1
Kurang(13) = 2
Kurang(14) = 2
Kurang(15) = 8
Kurang(16) = 8
syarat = 32
Banyaknya simpul yang dibangkitkan : 737
Banyaknya step yang ditempuh : 16
```

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 : 0.0440 sekon

input3.txt

Masukan nama file dengan ekstensinya (ex: test1.txt): input3.txt

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

Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 0
Kurang(4) = 0
Kurang(5) = 1
Kurang(6) = 4
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) = 8

syarat = 20

Banyaknya simpul yang dibangkitkan : 22
Banyaknya step yang ditempuh : 8

```

Down
-----
| 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 : 0.0030 sekon

D. Link Repository

https://github.com/buditato/Tucil3_13520132

| 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 | | ✓ |