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:

Kristo Abdi Wiguna 13520058

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

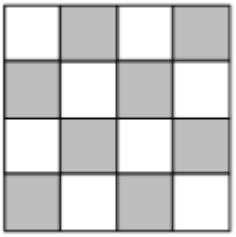
2022

DAFTAR ISI

Algoritma Branch And Bound

Algoritma BnB pada penyelesaian 15 puzzle akan meliputi class Puzzle yang dipanggil pertama kali setelah program sudah menerima matriks yang akan dipecahkan. Puzzle memiliki konstruktor yang terdiri dar 5 member yang berjenis dictionary yaitu tracker, linkedList, costDict, parentDict, dan depthDict. Tracker berguna untuk memasukkan tiap node masing – masing state matriks yang pernah dikunjungi untuk pruning langkah gerak tile kosong. Linked list berguna untuk menyimpan matriks simpul yang dieksplorasi sesuai gerak tile kosong. Untuk cost function c(i) = f(n) + h(n), costDict berguna untuk menyimpan taksiran jarak dari simpul ke goal state dengan menggunaka jumlah cost yang dimiliki tiap simpul yaitu jumlah letak tile yang tidak sesuai dengan goal state yaitu h(n). DepthDict berguna untuk menampung kedalaman atau jarak ke simpul dari akar yaitu f(n).

Pertama – tama, matriks akan ditentukan apakah dapat berjalan dengan Kurang(i) + X. Kurang(i) adalah jumlah dari tile yang memiliki posisi setelah nomor i yang kurang nilainya dari i. Paritas X adalah nilai biner tile kosong berada di state tile yang diarsir dibawah ini. Jika sesuai maka X bernilai satu dan sebaliknya nol. Jika nilai $\Sigma Kurang(i) + X$ genap, maka matriks bisa dipecahkan dan sebaliknya jika ganjil maka tidak dapat dipecahkan.



Ketika matrix dapat dipecahkan, maka lanjut ke langkah dimana matrix akan dijadikan dalam bentuk 1D array terlebih dahulu demi optimasi sistem. Matriks akan dihash sehingga untuk keperluan pruning tidak mengunjungi state matriks yang sama menjadi lebih cepat. Lalu, akan disimpan cost, depth, indeks root, dan state matriks di costDict, depthDict, parentDict dan linkedList. Algoritma branch and bound ingin memilih simpul dengan cost terkecil sehingga akan membuat child dari simpul tersebut, untuk itu digunakan priority queue yang menyimpan cost matriks pertama dan key simpul pertama.

Setelah itu, selama tidak ditemukan goal state dan priority queue masih berisi, loop akan dijalankan untuk mengenerate child dari simpul yang telah diambil dari priority queue dengan cost terminimum. Child yang digenerate adalah matriks dari simpul yang tile kosong digerakkan atas, bawah, kiri, dan kanan dan dimasukkan ke dalam directionMatrix jika pergerakan tile kosong valid dengan memanggil fungsi move. Ketika simpul dengan matriks pernah dikunjungi, maka pruning dilakukan dengan melewati simpul tersebut untuk tidak dikunjungi. Setelah sebuah simpul dikunjungi, maka cost dan depth akan dihitung dan state matriks simpul disimpan dalam tracker. Jika simpul yang sedang dieksplorasi adalah goal state, maka loop akan berhenti, jika bukan maka akan memasukkan simpul ke dalam priority queue. Hal ini dilakukan terus menerus hingga loop berhenti dan mendapat jumlah simpul yang dibangkitkan dan key dari goal node.

Source Program

Menggunakan bahasa pemrograman Python. Library yang digunakan adalah time, heapq, _pickle, dan random.

File main.py

```
import time
from node import *

def main() :
    state = "y"
    while (state == 'y' or state == 'Y') :
        print("============================================================")
        print("            15 PUZZLE SOLVER BY 13520058 KRISTO            ")
        print("============================================================")
        print("Menu : ")
        choice = input("1. Random generate puzzle\n2. Input filename\n3. Exit\nChoose (1/2/3) : ")
        if (choice == '1') :
            matrix = generateMatrix()
        elif (choice == '2') :
            print()
            print("Input Filename Testcase Formatting : <namafile>.txt")
            filename = "test/" + input("Input filename (example: test.txt): ")
            matrix = inputFileMatrix(filename)
            if (matrix == "-") : exit()

        if (choice == '1' or choice == '2') :
            sigma = kurangSigma(matrix) + isReachable(matrix)
            print(f'X = {isReachable(matrix)}')
            print(f'Sigma Kurang(i) : {sigma}')
            print("============================================================", end="")
            # Step 2
            if (sigma % 2 == 0) :
                now = time.time()
                puzzle = Puzzle()
                puzzle.solve(matrix)
                print(f'Elapsed execution time : {time.time()-now} seconds.')
            else :
                print("Puzzle can't be solved, use another puzzle!")
            # root = TreeNode(None, matrix, tilekosong, initialCost, 0, None)
            state = input("Do you want to continue (y/n) ? ")
        else :
            break
```

```
    print("====================================")
    print("              GOODBYE!              ")
    print("====================================")

if __name__ == "__main__":
    main()
```

File node,py

```python
from heapq import heappop, heappush
from util import *
from inout import *

class PrioQueue:
    def __init__(self):
        self.heap = []
    # Insert new element
    def push(self, k):
        heappush(self.heap, k)
    # Remove minimum element
    def pop(self):
        return heappop(self.heap)
    # Check if the Queue is empty
    def isEmpty(self):
        if not self.heap:
            return True
        else:
            return False
```

```python
class Puzzle :
    def __init__(self) :
        self.tracker = {}
        self.linkedList = {}
        self.costDict = {}
        self.parentDict = {}
        self.depthDict = {}

    def solve(self, initialMatrix) :
        key = self.initialize(initialMatrix)
        count_simpul, idxGoal = self.branchAndBound(initialMatrix, key)
        self.printPathSolution(count_simpul, idxGoal)

    def branchAndBound(self, initialMatrix, key) :
        # Create Priority Queue
        pq = PrioQueue()
        pq.push((self.costDict[key], key))
        found = False

        while ((not pq.isEmpty()) and (not found)):
            cost, nodeIdx = pq.pop()

            explore = self.linkedList[nodeIdx]
            # Create child, move empty tile goes up, down, left, right if valid
            if (isSolution(explore)):
                # F.S. key is the index of the goal state
                found = True
                break

            directionMatrix = getDirectionList(explore)
            for matrix in directionMatrix:
                if (matrix == -1):
                    # Move is invalid (kena ujung)
                    continue
```

```
            shorted = hashed(flattenMatrix(matrix))
            if (shorted in self.tracker):
                # Continue if state of matrix already visited
                continue

            # New node
            key += 1
            # Add to tracker every time a state of matrix visited
            self.tracker[shorted] = 1
            # Add to linked list to store matrix
            self.linkedList[key] = matrix
            # Cost Function
            self.depthDict[key] = self.depthDict[nodeIdx] + 1
            self.costDict[key] = countMisplacedTiles(matrix) + self.depthDict[key]
            # Add index to dict of parent index
            self.parentDict[key] = nodeIdx
            if (isSolution(matrix)):
                # F.S. key is the index of the goal state
                found = True
                break
            pq.push((self.costDict[key], key))

        count_simpul = key
        return count_simpul, key
```

```
def printPathSolution(self, count_simpul, key) :
    path = key
    solution = []
    done = False
    while (not done):
        solution.append(path)
        if (path == 1): done = True
        # Get linked to parent of each node
        path = self.parentDict[path]
    solution = solution[::-1]
    listOfMatrixRootToGoal = {}
    for i in range(len(solution)):
        listOfMatrixRootToGoal[i] = self.linkedList[solution[i]]

    for i in range(len(listOfMatrixRootToGoal)):
        # Print from root to goal state
        print()
        print()
        printMatrix(listOfMatrixRootToGoal[i])
    print(f'\nRaised nodes : {count_simpul}')
```

```
def initialize(self, initialMatrix) :
    # Create initial node for hashing in dict
    shorted = hashed(flattenMatrix(initialMatrix))
    key = 1
    # Create Dictionary
    # To contain dictionary with hashed matrix as key and number as a value
    self.tracker[shorted] = 1
    # To contain dictionary with number as key point to a matrix
    self.linkedList[1] = initialMatrix
    # To contain depth and cost of each node
    self.costDict[key] = countMisplacedTiles(initialMatrix)
    # To contain list of parent of each node
    self.parentDict[key] = key
    # To store depth of each node
    self.depthDict[key] = 0

    return key
```

File util.py

```
import random
import _pickle as cPickle

def unflattenMatrix(matrix) :
    res = [[0 for i in range(4)] for i in range(4)]
    x = 0
    for i in range(4) :
        for j in range(4) :
            res[i][j] = matrix[x]
            x += 1
    return res

def flattenMatrix(matrix) :
    res = [0 for i in range(16)]
    x = 0
    for i in range(4) :
        for j in range(4) :
            res[x] = matrix[i][j]
            x += 1
    return res

def hashed(matrix):
    res = ""
    symbol = "abcDEF.;[/']?GHIJKL"
    for element in matrix:
        res += symbol[element]
    return res

def isReachable(matrix) :
    solution = [[0,1,0,1],[1,0,1,0],[0,1,0,1],[1,0,1,0]]
    for i in range(len(matrix)) :
        for j in range(len(matrix[i])) :
            if (solution[i][j] == 1) and (matrix[i][j] == 16) :
                return 1
    return 0
```

```python
def isSolution(matrix) :
    return (countMisplacedTiles(matrix) == 0)

def kurang(matrix, row, col, kurangList) :
    res = 0
    for i in range(row, len(matrix)) :
        start = col if (i == row) else 0
        for j in range(start, len(matrix[i])) :
            if (row == i and col == j) : continue
            if (matrix[row][col] > matrix[i][j]) :
                res += 1

    kurangList.append([matrix[row][col], res])
    return res

def kurangSigma(matrix) :
    res = 0
    kurangList = []
    for i in range(len(matrix)) :
        for j in range(len(matrix[i])) :
            res += kurang(matrix, i, j, kurangList)
    kurangList = sorted(kurangList,key=lambda x: (x[0]))
    for i in range(len(kurangList)) :
        print(f'Kurang({kurangList[i][0]}) = {kurangList[i][1]}')
    return res

def countMisplacedTiles(matrix):
    solution = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
    count = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if ((matrix[i][j]) and
                (matrix[i][j] != solution[i][j])):
                count += 1
    return count
```

```
def getTileKosong(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])) :
            if matrix[i][j] == 16 :
                return [i,j]


def move(matrix, idx, direction):
    res = cPickle.loads(cPickle.dumps(matrix, -1))
    x = idx[0]
    y = idx[1]
    if (direction == 0):
        # TOP
        if (x > 0) :
            res[x][y], res[x - 1][y] = res[x - 1][y], res[x][y]
            return res
        else :
            return -1
    elif (direction == 1):
        # BOTTOM
        if (x < 3) :
            res[x][y], res[x + 1][y] = res[x + 1][y], res[x][y]
            return res
        else:
            return -1
    elif (direction == 2):
        # LEFT
        if (y > 0):
            res[x][y], res[x][y - 1] = res[x][y - 1], res[x][y]
            return res
        else:
            return -1
    else :
        # RIGHT
        if (y < 3):
            res[x][y], res[x][y + 1] = res[x][y + 1], res[x][y]
            return res
        else:
            return -1

def getDirectionList(matrix) :
    directionMatrix = []
    tilekosong = getTileKosong(matrix)
    for i in range(4) :
        directionMatrix.append(move(matrix, tilekosong, i))
    return directionMatrix
```

File inout.py

```
import _pickle as cPickle

def inputFileMatrix(filename) :
    matrix = []
    try :
        with open(filename) as f:
            for line in f:
                nums = [int(n) for n in line.split()]
                matrix.append(nums)
        printMatrix(matrix)
    except  :
        print("File tidak ditemukan! Silahkan ulangi kembali!")
        return "-"
    return matrix

def generateMatrix() :
    matrix = []
    choices = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
    for i in range(4):
        arr = []
        for j in range(4):
            val = random.choice(choices)
            arr.append(val)
            choices.remove(val)
        matrix.append(arr)
    printMatrix(matrix)
    return matrix

def printMatrix(matrix):
    for i in range(len(matrix)) :
        for j in range(len(matrix[i])) :
            print("%d " % (matrix[i][j]), end = " ")
        print()
```

Berkas File Uji

File tc.txt

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

File tc2.txt

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

File tc3.txt

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

File tc4.txt

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

File tc5.txt

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

Screenshot Input dan Output

File tc.txt

```
============================================================
15 PUZZLE SOLVER BY 13520058 KRISTO
============================================================
1. Random generate puzzle
2. Input filename
3. Exit
Choose (1/2/3) : 2
Masukkan filename tes uji dengan format <namafile>.txt
Input filename (contoh: test.txt): tc.txt
1  2  3  4
5  6  16  8
9  10  7  11
13  14  15  12
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 0
Kurang(4) = 0
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 0
Kurang(8) = 1
Kurang(9) = 1
Kurang(10) = 1
Kurang(11) = 0
Kurang(12) = 0
Kurang(13) = 1
Kurang(14) = 1
Kurang(15) = 1
Kurang(16) = 9
X = 1
Sigma Kurang(i) : 16
============================================================
```

```
Simpul ke-1 :

1  2  3  4
5  6  16  8
9  10  7  11
13  14  15  12
Simpul ke-2 :

1  2  3  4
5  6  7  8
9  10  16  11
13  14  15  12
Simpul ke-3 :

1  2  3  4
5  6  7  8
9  10  11  16
13  14  15  12
Simpul ke-4 :

1  2  3  4
5  6  7  8
9  10  11  12
13  14  15  16
Jumlah simpul yang dibuat : 10
Waktu yang dibutuhkan untuk menyelesaikan puzzle : 0.005998134613037109 detik.
Apakah anda ingin melanjutkan (y/n) ? n
=====================================
GOODBYE!
=====================================
```

File tc2.txt

```
============================================================
15 PUZZLE SOLVER BY 13520058 KRISTO
============================================================
1. Random generate puzzle
2. Input filename
3. Exit
Choose (1/2/3) : 2
Masukkan filename tes uji dengan format <namafile>.txt
Input filename (contoh: test.txt): tc2.txt
16  3  4  8
2  1  10  7
5  11  6  15
9  13  12  14
Kurang(1) = 0
Kurang(2) = 1
Kurang(3) = 2
Kurang(4) = 2
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 2
Kurang(8) = 5
Kurang(9) = 0
Kurang(10) = 4
Kurang(11) = 2
Kurang(12) = 0
Kurang(13) = 1
Kurang(14) = 0
Kurang(15) = 4
Kurang(16) = 15
X = 0
Sigma Kurang(i) : 38
============================================================
```

```
Simpul ke-1 :

16  3  4  8
2  1  10  7
5  11  6  15
9  13  12  14
Simpul ke-2 :

2  3  4  8
16  1  10  7
5  11  6  15
9  13  12  14
Simpul ke-3 :

2  3  4  8
1  16  10  7
5  11  6  15
9  13  12  14
Simpul ke-4 :

2  3  4  8
1  10  16  7
5  11  6  15
9  13  12  14
Simpul ke-5 :

2  3  4  8
1  10  6  7
5  11  16  15
9  13  12  14
Simpul ke-6 :

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

```
Simpul ke-7 :

2  3  4  8
1  10  6  7
5  11  12  15
9  13  14  16
Simpul ke-8 :

2  3  4  8
1  10  6  7
5  11  12  16
9  13  14  15
Simpul ke-9 :

2  3  4  8
1  10  6  7
5  11  16  12
9  13  14  15
Simpul ke-10 :

2  3  4  8
1  10  6  7
5  16  11  12
9  13  14  15
Simpul ke-11 :

2  3  4  8
1  16  6  7
5  10  11  12
9  13  14  15
Simpul ke-12 :

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

```
Simpul ke-13 :

2  3  4  8
1  6  7  16
5  10  11  12
9  13  14  15
Simpul ke-14 :

2  3  4  16
1  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-15 :

2  3  16  4
1  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-16 :

2  16  3  4
1  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-17 :

16  2  3  4
1  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-18 :

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

```
Simpul ke-19 :

1  2  3  4
5  6  7  8
16  10  11  12
9  13  14  15
Simpul ke-20 :

1  2  3  4
5  6  7  8
9  10  11  12
16  13  14  15
Simpul ke-21 :

1  2  3  4
5  6  7  8
9  10  11  12
13  16  14  15
Simpul ke-22 :

1  2  3  4
5  6  7  8
9  10  11  12
13  14  16  15
Simpul ke-23 :

1  2  3  4
5  6  7  8
9  10  11  12
13  14  15  16
Jumlah simpul yang dibuat : 4489
Waktu yang dibutuhkan untuk menyelesaikan puzzle : 0.13263940811157227 detik.
Apakah anda ingin melanjutkan (y/n) ? n
=====================================
GOODBYE!
=====================================
```

File tc3.txt

```
============================================================
15 PUZZLE SOLVER BY 13520058 KRISTO
============================================================
1. Random generate puzzle
2. Input filename
3. Exit
Choose (1/2/3) : 2
Masukkan filename tes uji dengan format <namafile>.txt
Input filename (contoh: test.txt): tc3.txt
3  1  2  4
16  5  7  8
10  6  11  12
9  13  14  15
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 2
Kurang(4) = 0
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 1
Kurang(8) = 1
Kurang(9) = 0
Kurang(10) = 2
Kurang(11) = 1
Kurang(12) = 1
Kurang(13) = 0
Kurang(14) = 0
Kurang(15) = 0
Kurang(16) = 11
X = 1
Sigma Kurang(i) : 20
============================================================
```

```
Simpul ke-1 :

3  1  2  4
16  5  7  8
10  6  11  12
9  13  14  15
Simpul ke-2 :

3  1  2  4
5  16  7  8
10  6  11  12
9  13  14  15
Simpul ke-3 :

3  1  2  4
5  6  7  8
10  16  11  12
9  13  14  15
Simpul ke-4 :

3  1  2  4
5  6  7  8
16  10  11  12
9  13  14  15
Simpul ke-5 :

3  1  2  4
16  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-6 :

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

```
Simpul ke-7 :

1  16  2  4
3  6  7  8
5  10  11  12
9  13  14  15
Simpul ke-8 :

1  6  2  4
3  16  7  8
5  10  11  12
9  13  14  15
Simpul ke-9 :

1  6  2  4
16  3  7  8
5  10  11  12
9  13  14  15
Simpul ke-10 :

1  6  2  4
5  3  7  8
16  10  11  12
9  13  14  15
Simpul ke-11 :

1  6  2  4
5  3  7  8
9  10  11  12
16  13  14  15
Simpul ke-12 :

1  6  2  4
5  3  7  8
9  10  11  12
13  16  14  15
Simpul ke-13 :

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

```
Simpul ke-14 :

1  6  2  4
5  3  7  8
9  10  16  12
13  14  11  15
Simpul ke-15 :

1  6  2  4
5  3  16  8
9  10  7  12
13  14  11  15
Simpul ke-16 :

1  6  2  4
5  16  3  8
9  10  7  12
13  14  11  15
Simpul ke-17 :

1  16  2  4
5  6  3  8
9  10  7  12
13  14  11  15
Simpul ke-18 :

1  2  16  4
5  6  3  8
9  10  7  12
13  14  11  15
Simpul ke-19 :

1  2  3  4
5  6  16  8
9  10  7  12
13  14  11  15
Simpul ke-20 :

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

```
Simpul ke-21 :

1  2  3  4
5  6  7  8
9  10  11  12
13  14  16  15
Simpul ke-22 :

1  2  3  4
5  6  7  8
9  10  11  12
13  14  15  16
Jumlah simpul yang dibuat : 17166
Waktu yang dibutuhkan untuk menyelesaikan puzzle : 0.3904891014099121 detik.
Apakah anda ingin melanjutkan (y/n) ? n
=====================================
GOODBYE!
=====================================
```

File tc4.txt

```
============================================================
15 PUZZLE SOLVER BY 13520058 KRISTO
============================================================
1. Random generate puzzle
2. Input filename
3. Exit
Choose (1/2/3) : 2
Masukkan filename tes uji dengan format <namafile>.txt
Input filename (contoh: test.txt): tc4.txt
12  15  4  11
1  7  9  10
13  3  2  5
14  8  6  16
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 1
Kurang(4) = 3
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 4
Kurang(8) = 1
Kurang(9) = 5
Kurang(10) = 5
Kurang(11) = 9
Kurang(12) = 11
Kurang(13) = 5
Kurang(14) = 2
Kurang(15) = 13
Kurang(16) = 0
X = 0
Sigma Kurang(i) : 59
============================================================

Puzzle tidak bisa diselesaikan, gunakan puzzle lain!
Apakah anda ingin melanjutkan (y/n) ? n
====================================
GOODBYE!
====================================
```

File tc5.txt

```
============================================================
15 PUZZLE SOLVER BY 13520058 KRISTO
============================================================
1. Random generate puzzle
2. Input filename
3. Exit
Choose (1/2/3) : 2
Masukkan filename tes uji dengan format <namafile>.txt
Input filename (contoh: test.txt): tc5.txt
1  3  4  15
2  16  5  12
7  6  11  14
8  9  10  13
Kurang(1) = 0
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
X = 0
Sigma Kurang(i) : 37
============================================================

Puzzle tidak bisa diselesaikan, gunakan puzzle lain!
```

Link Kode Program

https://github.com/kristabdi/15puzzle-solver-bnb

Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi	V	
2. Program berhasil running	V	
3. Program dapat menerima input dan menuliskan output	V	
4. Luaran sudah benar untuk semua data uji	V	
5. Bonus dibuat		V