Laporan Tugas Kecil 3
IF2211 Strategi Algoritma
Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Disusun oleh :
Muhammad Zunan Alfikri (13518019)

INSTITUT TEKNOLOGI BANDUNG
2020

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Buatlah program dalam Python untuk menyelesaikan persoalan 15-Puzzle dengan menggunakan Algoritma Branch and Bound seperti pada materi kuliah. Nilai bound tiap simpul adalah penjumlahan cost yang diperlukan untuk sampai suatu simpul x dari akar, dengan taksiran cost simpul x untuk sampai ke goal. Taksiran cost yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir (goal state). Untuk semua instansiasi persoalan 15-puzzle, susunan akhir yang diinginkan sesuai dengan Gambar 1.

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

Gambar 1. Susunan Akhir persoalan 15-puzzle

Masukan: matriks yang merepresentasikan posisi awal suatu instansiasi persoalan 15-puzzle. Matriks dibaca dari berkas teks.

Program harus dapat menentukan apakah posisi awal suatu masukan dapat diselesaikan hingga mencapai susunan akhir, dengan mengimplementasikan fungsi Kurang(i) dan posisi ubin kosong di kondisi awal (X), seperti pada materi kuliah. Jika posisi awal tidak bisa mencapai susunan akhir, program akan menampilkan pesan tidak bisa diselesaikan,. Jika dapat diselesaikan, program dapat menampilkan urutan matriks rute (path) aksi yang dilakukan dari posisi awal ke susunan akhir. Sebagai contoh pada Gambar 2, matriks yang ditampilkan ke layar adalah matriks pada simpul 1, simpul 4, simpul 10 dan simpul 23.

Keluaran:

1. Matriks posisi awal.
2. Nilai dari fungsi Kurang (i) untuk setiap ubin tidak kosong pada posisi awal (nilai ini tetap dikeluarkan, baik persoalan bisa diselesaikan atau tidak bisa diselesaikan).
3. Nilai dari $\sum_{i=1}^{16} KURANG(i) + X$
4. Jika persoalan tidak dapat diselesaikan (berdasarkan hasil butir 2) keluar pesan.
5. Jika persoalan dapat diselesaikan(berdasarkan hasil butir 2), menampilkan urutan matriks seperti pada penjelasan sebelummnya.
6. Waktu eksekusi
7. Jumlah simpul yang dibangkitkan dalam pohon ruang status

BAB II
ALGORITMA BRANCH AND BOUND

Pada penerapan program Branch and Bound untuk menyelesaikan persoalan 15-Puzzle kali ini menggunakan struktur data Tree. Tree disini menggambarkan keadaan setiap simpul. Setiap simpul memiliki elemen matriks (untuk menyimpan pola puzzle setiap simpul), basis(koordinat puzzle kosong), path(jalur yang ditempuh untuk sampai simpul tersebut), costfunction, dan cabang w, a, s, d untuk pembangkitan simpul anak-anaknya.

Berikut ini algoritma brach and bound secara umum yang digunakan :

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai costfunction yang paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung costfunction j, dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

Dalam persoalan ini, costfunction yang digunakan adalah penjumlahan layer simpul terkait dengan banyaknya petak yang beda dari simpul terkait dengan simpul target. Dalam persoalan ini, setiap simpul masing masing mempunyai anak sebanyak 4 simpul. Yaitu simpul ekpansi jika petak kosong di geser ke atas, kiri, kanan, atau bawah. Petak kosong direpresentasikan sebagai angka 0.

BAB III
KODE PROGRAM

Berikut ini kode program untuk menyelesaikan persoalan 15-Puzzle dalam bahasa python :

```
#import depedencies
import copy
import time
from collections import deque

#kelas puzzle
class Puzzle:
    banyak_simpul = 0
    reference = [[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,0]]

    def __init__(self):
    #fungsi untuk konstruktor
        Puzzle.banyak_simpul += 1
        self.mat = [[0 for i in range(4)] for i in range(4)]
        self.basis = (0,0)
        self.path = ""
        self.costfunction = 0
        self.u = None
        self.l = None
        self.d = None
        self.r = None

    def countCostFunction(self):
    #fungsi untuk menghitung costfunction untuk tiap simpul
        count  = 0
        for i in range(4):
            for j in range(4):
                if (self.mat[i][j] != Puzzle.reference[i][j]):
                    count += 1
        return count;

    def generateUp(self):
    #fungsi untuk generate sumpul jika puzzle yang kosong digeser ke atas
        if (self.basis[0] - 1 >= 0):
            Puzzle.banyak_simpul += 1
            self.u = copy.deepcopy(self)
            self.u.mat[self.basis[0]][self.basis[1]] =
self.u.mat[self.basis[0] - 1][self.basis[1]]
            self.u.mat[self.basis[0] - 1][self.basis[1]] = 0
            self.u.basis = (self.basis[0] - 1 , self.basis[1])
            self.u.path  += "u"
            self.u.costfunction = self.u.countCostFunction() +
len(self.u.path)

    def generateLeft(self):
    #fungsi untuk generate simpul jika puzzle yang kosong digeser ke kiri
        if (self.basis[1] - 1 >= 0):
            Puzzle.banyak_simpul += 1
            self.l = copy.deepcopy(self)
            self.l.mat[self.basis[0]][self.basis[1]] =
self.l.mat[self.basis[0]][self.basis[1] - 1]
            self.l.mat[self.basis[0]][self.basis[1] - 1] = 0
            self.l.basis = (self.basis[0] , self.basis[1] - 1)
```

```
            self.l.path  += "l"
            self.l.costfunction = self.l.countCostFunction() +
len(self.l.path)

    def generateDown(self):
    #fungsi untuk generate simpul jika puzzle yang kosong digeser ke
bawah
        if (self.basis[0] + 1 <= 3):
            Puzzle.banyak_simpul += 1
            self.d = copy.deepcopy(self)
            self.d.mat[self.basis[0]][self.basis[1]] =
self.d.mat[self.basis[0] + 1][self.basis[1]]
            self.d.mat[self.basis[0] + 1][self.basis[1]] = 0
            self.d.basis = (self.basis[0] + 1 , self.basis[1])
            self.d.path  += "d"
            self.d.costfunction = self.d.countCostFunction() +
len(self.d.path)

    def generateRight(self):
    #fungsi untuk generate simpul jika puzzle yang kosong digeser ke
kanan
        if (self.basis[1] + 1 <= 3):
            Puzzle.banyak_simpul += 1
            self.r = copy.deepcopy(self)
            self.r.mat[self.basis[0]][self.basis[1]] =
self.r.mat[self.basis[0]][self.basis[1] + 1]
            self.r.mat[self.basis[0]][self.basis[1] + 1] = 0
            self.r.basis = (self.basis[0] , self.basis[1] + 1)
            self.r.path  += "r"
            self.r.costfunction = self.r.countCostFunction() +
len(self.r.path)

    def cetakPuzzle(self):
    #fungsi untuk mencetak puzzle
        for i in self.mat:
            for j in range(4):
                print(i[j], end = " ")
            print()
        print("Cost Function : ", self.costfunction)
        print("Basis : ", self.basis[0], self.basis[1])
        print("Path : ", self.path)
        print("Banyak simpul : ", Puzzle.banyak_simpul)

    def inputFromConsole(self):
    #fungsi untuk menerima input dari console
        for i in range(4):
            temp = str(input())
            self.mat[i] = temp.split()
            for j in range(4):
                self.mat[i][j] = int(self.mat[i][j])
                if (self.mat[i][j] == 0):
                    self.basis = (i,j)
                if (self.mat[i][j] != Puzzle.reference[i][j]):
                    self.costfunction += 1

    def readFile(self, filename):
    #fungsi untuk membaca dari file dan dimasukkan ke matriks
        f = open(filename, "r")
        for i in range(4):
            temp = f.readline()
```

```
            self.mat[i] = temp.split()
            for j in range(4):
                self.mat[i][j] = int(self.mat[i][j])
                if (self.mat[i][j] == 0):
                    self.basis = (i,j)
                if (self.mat[i][j] != Puzzle.reference[i][j]):
                    self.costfunction += 1
        f.close()


    def cetakBasis(self):
    #fungsi untuk mencetak basis
        print(self.basis[0] , self.basis[1])

    def isSolvable(self):
    #fungsi untuk cek apakah solvable atau tidak
        count = 0
        temp = []
        for i in range(4):
            for j in range(4):
                if (self.mat[i][j] == 0):
                    temp.append(16)
                else :
                    temp.append(self.mat[i][j])
        for i in range(len(temp)):
            for j in range(i+1, len(temp)):
                if (temp[i] > temp[j]):
                    count += 1
        if ((self.basis[0] + self.basis[1]) % 2 == 1):
            count += 1
        print("Kurangi(i) : ", count)
        if (count % 2 == 0):
            # print("Solvable")
            return True
        else :
            # print("Not Solvable")
            return False

    def isInputValid(self):
    #fungsi mengembalikan true jika puzzle yang dimasukkan valid
        valid = [0 for i in range(16)]
        for i in range(4):
            for j in range(4):
                if (self.mat[i][j] > 15 or self.mat[i][j] < 0):
                    return False
                else :
                    valid[self.mat[i][j]] += 1
        for i in valid:
            if (i == 0):
                return False
        return True

def isEqual(matA, matB):
#mengembalikan true jika matriks A dan B sama
    for i in range(4):
        for j in range(4):
            if (matA[i][j] != matB[i][j]):
                return False
    return True
```

```
def isContain(kandidat, mat):
#mengembalikan true jika dalam array kandidat terdapat matriks mat
    for i in kandidat:
        if (isEqual(i, mat)):
            return True
    return False

def isFinish(mat, reference):
#mengembalikan true jika mat mencapai target
    return isEqual(mat, reference)

def sortQueue(Q):
#fungsi untuk sort Queue Q agar memenuhi prio Que
    for i in range(len(Q)):
        for j in range(i+1, len(Q)):
            if (Q[i].costfunction > Q[j].costfunction):
                temp = Q[i]
                Q[i] = Q[j]
                Q[j] = temp

def solvePuzzle(Puzz, Q, kandidat):
#fungsi untukk mencari path
    current = Puzz
    kandidat.append(Puzz.mat)
    end = isFinish(current.mat, Puzzle.reference)
    while(not(end)):
        #generate simpul W
        current.generateUp()
        if (current.u != None):
            if (isContain(kandidat, current.u.mat)):
                Puzzle.banyak_simpul -= 1
                current.u = None
            else :
                kandidat.append(current.u.mat)
                Q.append(current.u)
                if (isFinish(current.u.mat, Puzzle.reference)):
                    current = current.u
                    break
        #generate simpul A
        current.generateLeft()
        if (current.l != None):
            if (isContain(kandidat, current.l.mat)):
                Puzzle.banyak_simpul -= 1
                current.l = None
            else :
                kandidat.append(current.l.mat)
                Q.append(current.l)
                if (isFinish(current.l.mat, Puzzle.reference)):
                    current = current.l
                    break
        #generate S
        current.generateDown()
        if (current.d != None):
            if (isContain(kandidat, current.d.mat)):
                Puzzle.banyak_simpul -= 1
                current.d = None
            else :
                kandidat.append(current.d.mat)
                Q.append(current.d)
                if (isFinish(current.d.mat, Puzzle.reference)):
```

```
                    current = current.d
                    break
        #generate D
        current.generateRight()
        if (current.r != None):
            if (isContain(kandidat, current.r.mat)):
                Puzzle.banyak_simpul -= 1
                current.r = None
            else :
                kandidat.append(current.r.mat)
                Q.append(current.r)
                if (isFinish(current.r.mat, Puzzle.reference)):
                    current = current.r
                    break
        sortQueue(Q)
        current = Q.popleft()
        # print(Puzzle.banyak_simpul)
    return current

def masukanInput(Puz):
#prosedur untuk meminta pilihan input
    print("Pilih opsi cara memasukkan puzzle : ")
    print("1. Masukan dari file")
    print("2. Masukan dari console")
    x = int(input("Masukkan pilihan : "))
    if (x == 1):
        filename = str(input("Masukkan path file : "))
        Puz.readFile(filename)
    else :
        print()
        print("Masukkan 4 baris matriks, pisahkan dengan spasi, contoh :
\n1 2 3 4\n5 6 7 8\n9 10 11 12\n13 14 15 0")
        Puz.inputFromConsole()

def cetakMatriks(mat):
#prosedur untuk mencetak matriks
    for i in range(4):
        for j in range(4):
            print(mat[i][j], end = " ")
        print()

def cetakPath(res, puz):
#prosedur untuk mencetak path
    mat = puz.mat
    basis = puz.basis
    path = res.path
    for i in path:
        if (i == "u"):
            print("Langkah : up")
            mat[basis[0]][basis[1]] = mat[basis[0] - 1][basis[1]]
            mat[basis[0] - 1][basis[1]] = 0
            basis = (basis[0] - 1, basis[1])
        elif (i == "l"):
            print("Langkah : left")
            mat[basis[0]][basis[1]] = mat[basis[0]][basis[1] - 1]
            mat[basis[0]][basis[1] - 1] = 0
            basis = (basis[0], basis[1] - 1)
        elif (i == "d"):
            print("Langkah : down")
            mat[basis[0]][basis[1]] = mat[basis[0] + 1][basis[1]]
```

```
            mat[basis[0] + 1][basis[1]] = 0
            basis = (basis[0] + 1, basis[1])
        elif (i == "r"):
            print("Langkah : right")
            mat[basis[0]][basis[1]] = mat[basis[0]][basis[1] + 1]
            mat[basis[0]][basis[1] + 1] = 0
            basis = (basis[0], basis[1] + 1)
        cetakMatriks(mat)

#Kamus
kandidat = []
Q = deque()
Puz = Puzzle()

#Algoritma
print("============ SELAMAT DATANG DI PROGRAM 15 Puzzle !
================")
print()
masukanInput(Puz)
print()
if (not Puz.isInputValid()):
    print("Input tidak valid, masukkan Puzzle lagi !")
    masukanInput(Puz)

print()

if (Puz.isSolvable()):
    print("Puzzle dapat diselesaikan !")
    now = time.time()
    Result = solvePuzzle(Puz, Q, kandidat)
    print("Waktu yand dibutuhkan : ", time.time()-now)
    print("Simpul yang di bangkitkan : ", Puzzle.banyak_simpul)
    print("Path : ", Result.path)
    print()
    cetak = str(input("Cetak langkah ? (y/n)"))
    if (cetak == "Y" or cetak == "y"):
         cetakPath(Result, Puz)
else :
    print("Puzzle tidak dapat diselesaikan")
```

BAB IV
EKSPERIMEN

4.1 Spesifikasi Perangkat Keras

- Prosesor : AMD Ryzen 5 2500U Quad Core
- Layar : 14 Inch
- Tipe Grafis : AMD Radeon Vega 8 Graphics
- Kapasitas Penyimpanan : HDD 1TB + 128 GB M.2 SSD
- Memory : 8 GB DDR4-2400 SDRAM (1 x 4 GB)

4.2 Hasil Tangkapan Layar untuk Persoalan 1

Puzzle input :

```
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12
```

Output :

```
============ SELAMAT DATANG DI PROGRAM 15 Puzzle ! ================

Pilih opsi cara memasukkan puzzle :
1. Masukan dari file
2. Masukan dari console
Masukkan pilihan : 1
Masukkan path file : test.txt

Input Puzzle :
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12
Kurangi(i) :  16
Puzzle dapat diselesaikan !
Waktu yand dibutuhkan :  0.006981611251831055
Simpul yang di bangkitkan :  10
Banyak langkah :  3
Path :  drd

Cetak langkah ? (y/n)y
Langkah : down
1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12
Langkah : right
1 2 3 4
5 6 7 8
9 10 11 0
13 14 15 12
Langkah : down
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
```

4.3 Hasil Tangkapan Layar untuk Persoalan 2

Puzzle input :

```
0  2  3  4
1  6  7  8
5  10 11 12
9  13 14 15
```

Output :

```
============ SELAMAT DATANG DI PROGRAM 15 Puzzle ! ================

Pilih opsi cara memasukkan puzzle :
1. Masukan dari file
2. Masukan dari console
Masukkan pilihan : 1
Masukkan path file : test2.txt

Input Puzzle :
0 2 3 4
1 6 7 8
5 10 11 12
9 13 14 15
Kurangi(i) :  24
Puzzle dapat diselesaikan !
Waktu yand dibutuhkan :  0.0039975643157958984
Simpul yang di bangkitkan :  12
Banyak langkah :  6
Path :  dddrrr
```

```
Cetak langkah ? (y/n)y
Langkah : down
1 2 3 4
0 6 7 8
5 10 11 12
9 13 14 15
Langkah : down
1 2 3 4
5 6 7 8
0 10 11 12
9 13 14 15
Langkah : down
1 2 3 4
5 6 7 8
9 10 11 12
0 13 14 15
```

```
Langkah : right
1 2 3 4
5 6 7 8
9 10 11 12
13 0 14 15
Langkah : right
1 2 3 4
5 6 7 8
9 10 11 12
13 14 0 15
Langkah : right
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
```

4.4 Hasil Tangkapan Layar untuk Persoalan 3

Puzzle Test :

```
6 5 2 4
9 1 3 8
10 0 7 15
13 14 12 11
```

Output :

```
============ SELAMAT DATANG DI PROGRAM 15 Puzzle ! ================

Pilih opsi cara memasukkan puzzle :
1. Masukan dari file
2. Masukan dari console
Masukkan pilihan : 1
Masukkan path file : test3.txt

Input Puzzle :
6 5 2 4
9 1 3 8
10 0 7 15
13 14 12 11
Kurangi(i) :  34
Puzzle dapat diselesaikan !
Waktu yand dibutuhkan :  9.643820762634277
Simpul yang di bangkitkan :  958
Banyak langkah :  17
Path :  luruldrurdddruldr
```

```
Cetak langkah ? (y/n)y
Langkah : left
6 5 2 4
9 1 3 8
0 10 7 15
13 14 12 11
Langkah : up
6 5 2 4
0 1 3 8
9 10 7 15
13 14 12 11
Langkah : right
6 5 2 4
1 0 3 8
9 10 7 15
13 14 12 11
Langkah : up
6 0 2 4
1 5 3 8
9 10 7 15
13 14 12 11
Langkah : left
0 6 2 4
1 5 3 8
9 10 7 15
13 14 12 11
```

```
Langkah : down
1 6 2 4
0 5 3 8
9 10 7 15
13 14 12 11
Langkah : right
1 6 2 4
5 0 3 8
9 10 7 15
13 14 12 11
Langkah : up
1 0 2 4
5 6 3 8
9 10 7 15
13 14 12 11
Langkah : right
1 2 0 4
5 6 3 8
9 10 7 15
13 14 12 11
Langkah : down
1 2 3 4
5 6 0 8
9 10 7 15
13 14 12 11
Langkah : down
1 2 3 4
5 6 7 8
9 10 0 15
13 14 12 11
Langkah : down
1 2 3 4
5 6 7 8
9 10 12 15
13 14 0 11
Langkah : right
1 2 3 4
5 6 7 8
9 10 12 15
13 14 11 0
```

```
Langkah : up
1 2 3 4
5 6 7 8
9 10 12 0
13 14 11 15
Langkah : left
1 2 3 4
5 6 7 8
9 10 0 12
13 14 11 15
Langkah : down
1 2 3 4
5 6 7 8
9 10 11 12
13 14 0 15
Langkah : right
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
```

4.5 Hasil Tangkapan Layar untuk Persoalan 4

Puzzle test :

```
1 9 4 8
3 2 6 7
13 10 11 12
14 5 15 0
```

Output :

```
============ SELAMAT DATANG DI PROGRAM 15 Puzzle ! ================

Pilih opsi cara memasukkan puzzle :
1. Masukan dari file
2. Masukan dari console
Masukkan pilihan : 1
Masukkan path file : test4.txt

Kurangi(i) :  25
Puzzle tidak dapat diselesaikan
```

Puzzle persoalan 4 tidak dapat diselesaikan karena fungsi kurangi(i) bernilai ganjil.

4.6 Hasil Tangkapan Layar untuk Persoalan 5

Puzzle test:

```
12 15 4 11
1 7 9 10
13 3 2 5
14 8 6 0
```

Output :

```
============ SELAMAT DATANG DI PROGRAM 15 Puzzle ! ================

Pilih opsi cara memasukkan puzzle :
1. Masukan dari file
2. Masukan dari console
Masukkan pilihan : 1
Masukkan path file : test5.txt

Kurangi(i) :  59
Puzzle tidak dapat diselesaikan
```

Puzzle persoalan 5 tidak dapat diseleseaikan karena fungsi kurangi(i) bernilai ganjil.

LAMPIRAN

Tabel :

Poin	Ya	Tidak
1. Program berhasil di kompilasi	✔	
2. Program berhasil di running	✔	
3. Program dapat menerima input dan menuliskan output	✔	
4. Luaran sudah benar untuk semua data uji	✔	