Laporan Tugas Kecil 3

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

Mata Kuliah IF2211 - Strategi Algoritma

Oleh :

Nama : Jundan Haris

NIM : 13520155

Kelas : 02

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021/2022

DAFTAR ISI

BAB I
ALGORITMA *BRANCH AND BOUND*

Algoritma *Branch and Bound*

Algoritma *branch and bound* biasanya digunakan untuk optimasi, baik meminimalkan atau memaksimalkan suatu fungsi objektif dan tidak melanggar Batasan (*constraint)* yang ada. Secara garis besar, algoritma *branch and bound* dapat dikatakan sebagai perpaduan algoritma BFS (*Breadth First Search*) dan *least cost search*. Pada BFS murni, simpul berikutnya yang akan diekspansi mengikuti urutan pembangkitannya yaitu FIFO (*First In First Out*). Sedangkan pada algoritma *branch and bound* setiap simpul diberi sebuah nilai *cost.* Pada algoritma ini, simpul yang akan diekspansi yaitu berdasarkan *cost* dari setiap simpulnya. Apabila pada persoalan minimasi, maka simpul dengan cost paling rendah terlebih dahulu yang akan dibangkitkan. Dan pada persoalan maksimasi, adalah sebaliknya.

Algoritma *Branch and Bound* dalam Pembuatan *15-Puzzle Solver*

15-Puzzle merupakan permainan strategi yang cukup popular. Cara bermainnya cukup sederhana, yaitu terdapat sepetak ubin dengan ukuran $4 \, x \, 4$ yang berisikan 15 angka yang teracak dan satu ubin kosong. Tujuan permainan adalah untuk mencapai *goalstate* dengan cara menggeser ubin kosong ke atas, ke kanan, ke bawah, atau ke kiri. *Goalstate* yang dimaksud adalah ketika semua angka terurut dan ubin kosong berada di ubin paling bawah sebelah kanan.

Untuk menyelesaikan permainan tersebut, dapat digunakan algoritma *branch and bound* yang diimplementasikan menggunakan bahasa pemrograman, salah satunya adalah python. Secara umum, langkah-langkah penyelesaiannya adalah :

1. Baca file masukan dengan format .txt yang berisikan state awal permainan.
2. Hitung nilai $\sum_{i=1}^{16} KURANG(i) + X$.
3. Apabila hasilnya ganjil, maka puzzle tidak dapat diselesaikan. Apabila hasilnya genap, maka puzzle tersebut dapat diselesaikan.
4. Setelah itu cek terlebih dahulu apakah matriks/puzzle yang sekarang adalah goal state, apabila iya maka selesai, apabila tidak maka lanjut ke langkah selanjutnya.
5. Buat node baru dengan menggunakan matriks masukan.
6. Masukkan node tersebut ke dalam antrian priorityqueue.
7. Apabila antrian priorityqueue kosong atau goal state sudah ditemukan, maka berhenti.

8. Jika priorityqueue tidak kosong, pilih node dengan cost paling kecil. Apabila terdapat lebih dari satu simpul, ikuti aturan FIFO. Penghitungan cost pada algoritma ini menggunakan rumus

$$c(n) = f(n) + g(n)$$

Dengan n adalah node, $c(n)$ merupakan cost dari node n, $f(n)$ adalah Panjang lintasan dari root ke node atau kedalaman node n, dan $g(n)$ adalah jumlah tiles yang tidak sesuai dengan posisinya (goal state).

9. Jika node yang diambil adalah goal state, maka pencarian berhenti. Apabila bukan, maka lanjut ke langkah selanjutnya.
10. Bangkitkan seluruh anak dari node yang diambil. Lalu masukkan ke dalam antrian priorityqueue. Hitung cost dari setiap anak. Cost tersebut akan menentukan posisi anak pada antrian.
11. Ulangi langkah ke-7 hingga goal state ditemukan.

BAB II
SOURCE PROGRAM DALAM BAHASA PYTHON

Pada pengerjaannya, program ditulis dalam bahasa python. Program secara lengkap dapat diakses pada pranala https://github.com/jundanha/Tucil3_13520155 . Pada bab ini akan ditampilkan beberapa bagian potongan program guna memenuhi spesifikasi dari Tugas Kecil 3 IF2211 Strategi Algoritma.

Node.py

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

Prioqueue.py

```
class PriorityQueue:
    # Constructor
    def __init__(self, priority_function):
        self.queue = []
        self.func = priority_function

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

    # Add item to PQ
    def enqueue(self, item):
        pos = 0
        found = False

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

        self.queue.insert(pos, item)

    # Remove item from PQ
    def dequeue(self):
        return self.queue.pop(0)
```

Solver.py

```
#import
import time
from node import *
from prioqueue import *
import copy

#membaca file dari folder testcase
def readfile(dir):
    f = open(dir, 'r')
    lines = f.readlines()
    matrix = []
    #masukkan ke dalam matrix
    for line in lines:
        temp = line.strip("\n").split(" ")
        matrix.append(temp)
    for i in range(4):
        for j in range(4):
            if matrix[i][j] == '-':
                matrix[i][j] = 16
            else :
                matrix[i][j] = int(matrix[i][j])
    return matrix

#menampilkan matrix puzzle
def printmatrix(matrix):
    for i in range(4):
        for j in range(4):
            if (matrix[i][j] == 16):
                print("-", end="  ")
            else:
                if (matrix[i][j] >= 10):
                    print(matrix[i][j], end=" ")
                else :
                    print(matrix[i][j], end="  ")
        print()

#mencetak tabel kurang(i)
def printkurangi(matrix):
    print("i : kurang(i)")
    for i in range(1, 17):
        x, y = findidx(matrix, i)
        kurangi = hitungkurangi(matrix, x, y)
        print(i, ":", kurangi)

#mencari indeks dari number di matrix
def findidx(matrix, i):
    for x in range(4):
```

```
        for y in range(4):
            if matrix[x][y] == i:
                return x, y

#menghitung kurang(i) dari posisi x, y
def hitungkurangi(matrix, x, y):
    count = 0
    number = matrix[x][y]
    for i in range(x, 4):
        for j in range(4):
            if (i == x and j < y):
                continue
            else :
                if (matrix[i][j] < number):
                    count += 1
    return count

#menghitung sigma kurang(i)
def sigmakurangi(matrix):
    sigma = 0
    for i in range(1, 17):
        x, y = findidx(matrix, i)
        kurangi = hitungkurangi(matrix, x, y)
        sigma += kurangi
    return sigma

#menentukan X adalah 0 atau 1
def isXone(matrix):
    x, y = findidx(matrix, 16)
    if ((x % 2 == 0 and y % 2 != 0) or (x % 2 != 0 and y % 2 == 0)):
        return 1
    return 0

#memeriksa apakah matrix sudah solved
def isSolved(matrix):
    idx = 1
    for i in range(4):
        for j in range(4):
            if (matrix[i][j] != idx):
                return False
            idx += 1
    return True

#menghitung jumlah indeks yang tidak sesuai
def countWrongPosition(matrix):
    count = 0
    number = 1
    for i in range(4):
```

```
        for j in range(4):
            if (matrix[i][j] != number):
                count += 1
            number += 1
    return count

#mencari semua possible move
def findpossiblemove(node):
    x, y = findidx(node.matrix, 16)
    temp = []
    if (x > 0):
        temp.append("up")
    if (x < 3):
        temp.append("down")
    if (y > 0):
        temp.append("left")
    if (y < 3):
        temp.append("right")
    return temp

#menggeser ubin kosong
def movetiles(matrix, move):
    x, y = findidx(matrix, 16)
    temp = matrix[x][y]
    if move == "up":
        matrix[x][y] = matrix[x - 1][y]
        matrix[x - 1][y] = temp
    elif move == "right":
        matrix[x][y] = matrix[x][y + 1]
        matrix[x][y + 1] = temp
    elif move == "down":
        matrix[x][y] = matrix[x + 1][y]
        matrix[x + 1][y] = temp
    elif move == "left":
        matrix[x][y] = matrix[x][y - 1]
        matrix[x][y - 1] = temp
    return matrix

#melihat matriks apabila digeser dengan arah move
def seemovetiles(matrix, move):
    x, y = findidx(matrix, 16)
    temp = copy.deepcopy(matrix)
    if move == "up":
        temp[x][y] = temp[x - 1][y]
        temp[x - 1][y] = 16
    elif move == "right":
        temp[x][y] = temp[x][y + 1]
        temp[x][y + 1] = 16
```

```
    elif move == "down":
        temp[x][y] = temp[x + 1][y]
        temp[x + 1][y] = 16
    elif move == "left":
        temp[x][y] = temp[x][y - 1]
        temp[x][y - 1] = 16
    return temp

#menampilkan solusi secara step by step
def showSolution(node, matrix):
    move = []
    while (node.parent != None):
        move.append(node.move)
        node = node.parent
    idx = 1
    if move == []:
        print("Puzzle merupakan solusi!")
    for i in range(len(move)-1, -1, -1):
        print("MOVE "+ str(idx), ": ", end="")
        idx += 1
        temp = move[i]
        if (temp == "up"):
            print("UP")
        elif (temp == "down"):
            print("DOWN")
        elif (temp == "left"):
            print("LEFT")
        elif (temp == "right"):
            print("RIGHT")
        printmatrix(movetiles(matrix, move[i]))
        print()
    print()
    print("Jumlah step =", len(move))

#main program
print("======= 15-Puzzle Solver =======\n")
filename = input("Masukkan nama file: ")
filefound = True
try :
    matrix = readfile("testcase/" + filename)
except:
    print("file tidak ditemukan")
    filefound = False

if filefound:
    print("\n=========== Puzzle ===========")
    printmatrix(matrix)
    print("\n====== Tabel KURANG(i) =======")
```

```
    printkurangi(matrix)
    print("sigma kurang(i) =", sigmakurangi(matrix))
    print("nilai X =", isXone(matrix))
    temp = sigmakurangi(matrix) + isXone(matrix)
    print("sigma kurang(i) + X =", temp)

    #Puzzle tidak dapat diselesaikan
    if (temp % 2 != 0):
        print("\nGoal state tidak dapat dicapai")

    #Puzzle dapat diselesaikan
    else :

        #Algoritma Branch and Bound
        print("\nGoal state dapat dicapai")
        print("solving puzzle...")

        #untuk menghitung waktu eksekusi
        start = time.time()

        #inisialisasi
        prioq = PriorityQueue(lambda a,b : a.cost < b.cost)
        nodecount = 1
        depth = 0
        statevisited = {}
        nodesolution = None

        finished = isSolved(matrix)
        #apabila matriks awal adalah matriks goal
        if finished:
            statevisited[str(matrix)] = True
            nodesolution = node()
            nodesolution.matrix = matrix
            nodesolution.parent = None
        #apabila matriks awal bukan matriks goal
        else :
            #inisialisasi root node
            rootNode = node()
            rootNode.matrix = matrix
            rootNode.cost = countWrongPosition(matrix) + depth
            rootNode.depth = depth
            rootNode.parent = None
            rootNode.move = ""

            prioq.enqueue(rootNode)

            #loop
            while((not finished) or (not prioq.is_empty)):
```

```
            temp = prioq.dequeue()

            #memeriksa matriks
            if (isSolved(temp.matrix)):
                nodesolution = temp
                finished = True
                continue
            statevisited[str(temp.matrix)] = True

            for move in findpossiblemove(temp):
                #memasukkan state-state baru ke prioqueue
                moved = seemovetiles(temp.matrix, move)
                if (str(moved) not in statevisited):
                    temp1 = node()
                    temp1.matrix = moved
                    temp1.depth = temp.depth + 1
                    temp1.cost = countWrongPosition(moved) + temp1.depth
                    temp1.parent = temp
                    temp1.move = move

                    prioq.enqueue(temp1)
                    nodecount += 1

    #menampilkan solusi
    print("solved!\n")
    print(">> SOLUTION STEP BY STEP\n")
    showSolution(nodesolution, matrix)
    print("jumlah node =", nodecount)
    end = time.time()
    print("Waktu yang dibutuhkan : " + str(end-start) + " detik")
    print()
```

Daftar Fungsi

No	Fungsi	Parameter	Keterangan
1	readfile	dir	Membaca file dan memasukkan puzzle ke dalam matriks
2	printmatrix	matrix	Menampilkan matriks puzzle
3	printkurangi	matrix	Menampilkan tabel kurang(i)
4	findidx	matrix, i	Mencari indeks baris dan kolom dari i pada matriks matrix
5	hitungkurangi	matrix, x, y	Menghitung nilai kurang(i) elemen matrix dengan indeks baris x dan indeks kolom y
6	sigmakurangi	matrix	Menghitung total nilai dari kurang(i) pada sebuah matriks matrix
7	isXone	matrix	Menentukan apakan tiles kosong memiliki nilai X = 0 atau X =1

8	isSolved	matrix	Memeriksa apakah matrix sudah sesuai dengan goal state
9	countWrongPosition	matrix	Menghitung berapa jumlah tiles yang tidak sesuai dengan posisi yang seharusnya (posisi goal state)
10	findpossiblemove	node	Mencari seluruh move yang bisa dilakukan oleh sebuah node
11	movetiles	matrix, move	Menggeser ubin kosong dengan arah move
12	seemovetiles	matrix, move	Melihat matriks apabila digeser dengan arah move (tidak mengubah posisi matrix, hanya melihat saja)
13	showSolution	node, matrix	Menampilkan solusi puzzle dengan memperlihatkan setiap *step*-nya

BAB III
CONTOH *INPUT* DAN *OUTPUT* PROGRAM

Testcase1 (*Not Solvable*)

src > testcase > tc1.txt

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

```
PS D:\coolyeah\semester 4\IF2211 Strategi Algoritma\Tugas\Tucil3_13520155\src> py solver.py
======= 15-Puzzle Solver =======

Masukkan nama file: tc1.txt

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

====== Tabel KURANG(i) =======
i : kurang(i)
1 : 0
2 : 1
3 : 0
4 : 1
5 : 0
6 : 4
7 : 2
8 : 1
9 : 0
10 : 5
11 : 8
12 : 8
13 : 12
14 : 2
15 : 6
16 : 4
sigma kurang(i) = 54
nilai X = 1
sigma kurang(i) + X = 55

Goal state tidak dapat dicapai
```

Testcase2 (*Not Solvable*)

src > testcase > tc2.txt

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

```
PS D:\coolyeah\semester 4\IF2211 Strategi Algoritma\Tugas\Tucil3_13520155\src> py solver.py
======= 15-Puzzle Solver =======

Masukkan nama file: tc2.txt

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

====== Tabel KURANG(i) =======
i : 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
sigma kurang(i) = 37
nilai X = 0
sigma kurang(i) + X = 37

Goal state tidak dapat dicapai
```

Testcase3 (*Solvable*)

```
src > testcase > ≡ tc3.txt
1    1 2 3 4
2    5 6 7 8
3    9 10 11 12
4    13 14 15 -
```

```
PS D:\coolyeah\semester 4\IF2211 Strategi Algoritma\Tugas\Tucil3_13520155\src> py solver.py
======= 15-Puzzle Solver =======

Masukkan nama file: tc3.txt

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

====== Tabel KURANG(i) =======
i : kurang(i)
1 : 0
2 : 0
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
sigma kurang(i) = 0
nilai X = 0
sigma kurang(i) + X = 0

Goal state dapat dicapai
solving puzzle...
solved!

>> SOLUTION STEP BY STEP

Puzzle merupakan solusi!

Jumlah step = 0
jumlah node = 1
Waktu yang dibutuhkan : 0.0 detik
```

Testcase4 (*Solvable*)

src > testcase > tc4.txt

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

```
PS D:\coolyeah\semester 4\IF2211 Strategi Algoritma\Tugas\Tucil3_13520155\src> py solver.py
======= 15-Puzzle Solver =======

Masukkan nama file: tc4.txt

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

====== Tabel KURANG(i) =======
i : kurang(i)
1 : 0
2 : 0
3 : 1
4 : 1
5 : 4
6 : 0
7 : 1
8 : 1
9 : 4
10 : 0
11 : 1
12 : 0
13 : 2
14 : 1
15 : 5
16 : 7
sigma kurang(i) = 28
nilai X = 0
sigma kurang(i) + X = 28

Goal state dapat dicapai
solving puzzle...
solved!

>> SOLUTION STEP BY STEP

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

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

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

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

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

MOVE 6 : DOWN
13 14 15 12

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

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

Jumlah step = 10
jumlah node = 25
Waktu yang dibutuhkan : 0.014999151229858398 detik
```

Testcase5 (*Solvable*)

src > testcase > tc5.txt

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

```
PS D:\coolyeah\semester 4\IF2211 Strategi Algoritma\Tugas\Tucil3_13520155\src> py solver.py
======= 15-Puzzle Solver =======

Masukkan nama file: tc5.txt

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

====== Tabel KURANG(i) =======
i : kurang(i)
1 : 0
2 : 0
3 : 0
4 : 1
5 : 1
6 : 4
7 : 1
8 : 0
9 : 0
10 : 0
11 : 1
12 : 0
13 : 2
14 : 2
15 : 5
16 : 5
sigma kurang(i) = 22
nilai X = 0
sigma kurang(i) + X = 22

Goal state dapat dicapai
solving puzzle...
solved!
>> SOLUTION STEP BY STEP

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

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

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

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

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

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

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

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

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

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

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

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

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

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


MOVE 17 : DOWN
1  2  3  4
5  6  -  8
9  10 7  11
13 14 15 12

MOVE 18 : DOWN
1  2  3  4
5  6  7  8
9  10 -  11
13 14 15 12

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

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


Jumlah step = 20
jumlah node = 25792
Waktu yang dibutuhkan : 42.98655557632446 detik
```

BAB IV
KESIMPULAN DAN SARAN

Kesimpulan

Program 15-Puzzle Solver dengan mengimplementasikan algoritma brach and bound dapat diselesaikan dengan baik. Program juga dapat berjalan dengan baik karena dapat memenuhi semua spesifikasi tugas yang diberikan. Namun spesifikasi bonus tidak dapat dibuat oleh penulis. Selain itu mungkin saja masih ada beberapa kesalahan yang tidak saya sadari.

Berikut adalah tabel yang diperlukan untuk mempermudah penilaian.

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		✓

Saran

Program yang penulis buat sangat jauh dari kata sempurna, masih banyak yang bisa diperbaiki dan diperbagus lagi. Penulis juga tidak sempat mengerjakan spesifikasi bonus yang diberikan. Selama pengerjaan tugas ini, penulis juga banyak menemukan kesulitan. Untuk kedepannya, mungkin dapat diselesaikan dengan lebih cepat dan lebih baik apabila mendengarkan kelas dengan lebih seksama, mempelajari materi lebih jauh tidak terbatas ketika kelas, mengerjakan tugas di awal waktu.

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm

https://www.geeksforgeeks.org/python-string-strip-2/#:~:text=The%20strip()%20method%20in,be%20removed%20from%20the%20string.

https://www.programiz.com/python-programming/matrix

https://www.w3schools.com/python/python_file_open.asp

https://www.w3schools.com/python/python_for_loops.asp

https://www.geeksforgeeks.org/copy-python-deep-copy-shallow-copy/#:~:text=Deep%20copy%20is%20a%20process,is%20copied%20in%20other%20object.

https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/

https://www.geeksforgeeks.org/priority-queue-in-python/

https://www.geeksforgeeks.org/priority-queue-using-queue-and-heapdict-module-in-python/