TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA
Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Dibuat oleh:
Rizky Akbar Asmaran
13520111

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

A. Algoritma Branch and Bound

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

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

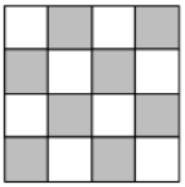
Permainan 15-Puzzle ini mempunyai 16! Atau sekitar 20.9 x 10^{12} susunan ubin yang berbeda dan hanya setengahnya yang dapat dicapai dari state awal sembarang. Akan tetapi, sebelum menelusuri ruang status untuk mencapai solusi susunan ubin, kita perlu memeriksa apakah akhir ubin dapat dicapai dari status awal. Status tujuan dapat dicapai apabila $\sum_{i=1}^{16} kurang(i) + X$ bernilai genap. Fungsi kurang (i) ini merupakan banyaknya ubin bernomor j sehingga j < I dan posisi(j) > posisi(i). Posisi(i) ini merupakan posisi ubin I pada susuan yang diperiksa. Sedangkan nilai X ditentukan bedasarkan posisi awal ubin kosong. Jika ubin kosong terdapat pada sel yang diarsir, maka nilai X = 1, sebaliknya jika ubin kosong tidak berada pada sel yang diarsir, nilai X = 0.

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

State awal

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

State akhir



Apabila kita sudah mengetahui bahwa susunan puzzle merupakan susunan yang solveable (dapat diselesaikan), maka kita dapat melanjutkan Langkah berikutnya, yaitu menelusuri ruang status hingga mencapai susunan ubin yang merupakan solusi menggunakan algoritma *Branch and Bound*. Ruang status ubin ini direpresentasikan dengan menggunakan konsep pohon. Anak dari setiap simpul P pada pohon menyatakan status yang dapat dicapai dari status P. perpindahan antar simpul dinyatakan sebagai perpindahan ubin kosong ke atas, kanan, bawah, ataukiri. Dengan menggunakan algoritma *Branch and Bound*, setiap simpul P pada pohon ruang status memiliki sebuah nilai *cost* c(P) yang menyatakan nilai batas (*bound*) untuk simpul tersebut. *Cost* untuk simpul ini biasanya berupa taksiran

$$\hat{c}(i) = f(i) + \hat{g}(i)$$

$\hat{c}$ (i) = ongkos untuk simpul I, $f(i)$ = ongkos untuk mencapai simpul i dari akar, $\hat{g}(i)$ = ongkos untuk mencapai simpul tujuan (*goal node*) dari simpul i

Untuk persoalan 15-Puzzle secara spesifik untuk simpul P

$f(P)$ = panjang lintasan dari simpul akar ke simpul P

$\hat{g}(P)$ = taksiran Panjang lintasan terpendek dari P ke simpul solusi pada upapohon yang akarnya P, yaitu berupa jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.

Proses pencarian yang dilakukan dengan algoritma *Branch and Bound* ini dibantu dengan penggunaan prinsip *least cost search* pada pencarian simpul anak pada tiap langkahnya, *least cost search* pada pencarian simpul anak pada tiap langkahnya. *Least cost search* digunakan untuk menentukan simpul child mana yang ditelusuri selanjutnya. Tiap pembangkitan simpul child, nantinya akan dihitung nilai *costnya* dan nantinya akan dimasukkan kedalam antrian pemrosesan simpul hidup. Untuk penggunaaan Queue digunakan priority queue yang memiliki prioritas utama yang simpul childnya adalah nilai *cost* terkecil. Oleh karena itu, simpul child yang dipilih memiliki cost terkecil dari simpul child yang lain pada antrian.

Langkah Algoritma *Branch and Bound* untuk menyelesaikan persoalan ini adalah:

1. Memasukkan simpul akar ke dalam antrian (*queue*) Q. Apabila simpul akar adalah simpul solusi (*goal node*), ini menyatakan solusi berhasil ditemukan. Oleh karena itu, penelusuran berhenti.
2. Apabila Q kosong, ini menyatakan bahwa tidak ditemukan solusi didalam persoalan tersebut. Oleh karena itu, penelusuran berhenti.
3. Apabila Q tidak kosong. Pilih dari antrian (*queue*) Q simpul i yang mempunyai nilai cost yang paling kecil. Apabila terdapat beberapa simpul i yang memenuhi, pilih salah satu secara *random*.
4. Apabila simpul i merupakan simpul solusi, ini menyatakan bahwa solusi sudah ditemukan. Lalu, penelusuran diberhentikan. Apabila simpul i bukan merupakan simpul solusi, pohon akan membangkitkan semua anak anaknya.
5. Untuk setiap anak j dari simpul i, hitung costnya dan masukkan semua anak anak tersebut ke dalam Q.
6. Ulangi Langkah-langkah diatas mulai dari Langkah kedua.

B. Screenshot input-output program

1. Uji Test 1

Puzzle1.txt

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

```
====================
= 15-Puzzle-Solver =
====================
Pilih Metode:
1. Input File
2. Randomizer
3. Exit
>> 1
Masukkan nama file : test/puzzle1.txt
=============
|1 |2 |3 |4 |
=============
|5 |6 |8 |11|
=============
|9 |10|7 |X |
=============
|13|14|15|12|
=============
ubin ke-1 : 0
ubin ke-2 : 0
ubin ke-3 : 0
ubin ke-4 : 0
ubin ke-5 : 0
ubin ke-6 : 0
ubin ke-7 : 0
ubin ke-8 : 0
ubin ke-9 : 1
ubin ke-10 : 1
ubin ke-11 : 1
ubin ke-12 : 1
ubin ke-13 : 1
ubin ke-14 : 1
ubin ke-15 : 3
ubin ke-16 : 4
Nilai fungsi kurang untuk tiap ubin:
Total :  13
Total Kurang(i) + X =  14
Status:
Puzzle bisa diselesaikan
```

```
>> Atas
=============
|1 |2 |3 |4 |
=============
|5 |6 |8 |X |
=============
|9 |10|7 |11|
=============
|13|14|15|12|
=============
>> Kiri
=============
|1 |2 |3 |4 |
=============
|5 |6 |X |8 |
=============
|9 |10|7 |11|
=============
|13|14|15|12|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|X |11|
=============
|13|14|15|12|
=============
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|X |
=============
|13|14|15|12|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|12|
=============
|13|14|15|X |
=============
Jumlah simpul yang dibangkitkan : 17
waktu : 0.015244499999999661 detik
Apakah anda ingin mencoba lagi? (y/n) :
```

2. Uji test 2 Puzzle2.txt 1 2 3 4 5 16 6 8 9 10 7 12 13 14 11 15	``` ==================== = 15-Puzzle-Solver = ==================== Pilih Metode: 1. Input File 2. Randomizer 3. Exit >> 1 Masukkan nama file : test/puzzle2.txt ============= |1 |2 |3 |4 | ============= |5 |X |6 |8 | ============= |9 |10|7 |12| ============= |13|14|11|15| ============= ubin ke-1 : 0 ubin ke-2 : 0 ubin ke-3 : 0 ubin ke-4 : 0 ubin ke-5 : 0 ubin ke-6 : 0 ubin ke-7 : 0 ubin ke-8 : 0 ubin ke-9 : 0 ubin ke-10 : 1 ubin ke-11 : 1 ubin ke-12 : 1 ubin ke-13 : 1 ubin ke-14 : 1 ubin ke-15 : 1 ubin ke-16 : 10 Nilai fungsi kurang untuk tiap ubin: Total : 16 Total Kurang(i) + X = 16 Status: Puzzle bisa diselesaikan ```

```
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |X |8 |
=============
|9 |10|7 |12|
=============
|13|14|11|15|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|X |12|
=============
|13|14|11|15|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|12|
=============
|13|14|X |15|
=============
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|12|
=============
|13|14|15|X |
=============
Jumlah simpul yang dibangkitkan : 12
waktu : 0.014660300000002735 detik
Apakah anda ingin mencoba lagi? (y/n) :
```

3. Uji test 3 Puzzle3.txt 1 2 3 4 6 10 7 8 5 13 15 11 9 16 14 12	``` ==================== = 15-Puzzle-Solver = ==================== Pilih Metode: 1. Input File 2. Randomizer 3. Exit >> 1 Masukkan nama file : test/puzzle3.txt ============= |1 |2 |3 |4 | ============= |6 |10|7 |8 | ============= |5 |13|15|11| ============= |9 |X |14|12| ============= ubin ke-1 : 0 ubin ke-2 : 0 ubin ke-3 : 0 ubin ke-4 : 0 ubin ke-5 : 0 ubin ke-6 : 0 ubin ke-7 : 0 ubin ke-8 : 1 ubin ke-9 : 1 ubin ke-10 : 1 ubin ke-11 : 1 ubin ke-12 : 1 ubin ke-13 : 2 ubin ke-14 : 3 ubin ke-15 : 4 ubin ke-16 : 4 Nilai fungsi kurang untuk tiap ubin: Total : 18 Total Kurang(i) + X = 18 Status: Puzzle bisa diselesaikan ```

```
>> Atas
=============
|1 |2 |3 |4 |
=============
|6 |10|7 |8 |
=============
|5 |X |15|11|
=============
|9 |13|14|12|
=============
>> Atas
=============
|1 |2 |3 |4 |
=============
|6 |X |7 |8 |
=============
|5 |10|15|11|
=============
|9 |13|14|12|
=============
>> Kiri
=============
|1 |2 |3 |4 |
=============
|X |6 |7 |8 |
=============
|5 |10|15|11|
=============
|9 |13|14|12|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|X |10|15|11|
=============
|9 |13|14|12|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|15|11|
```

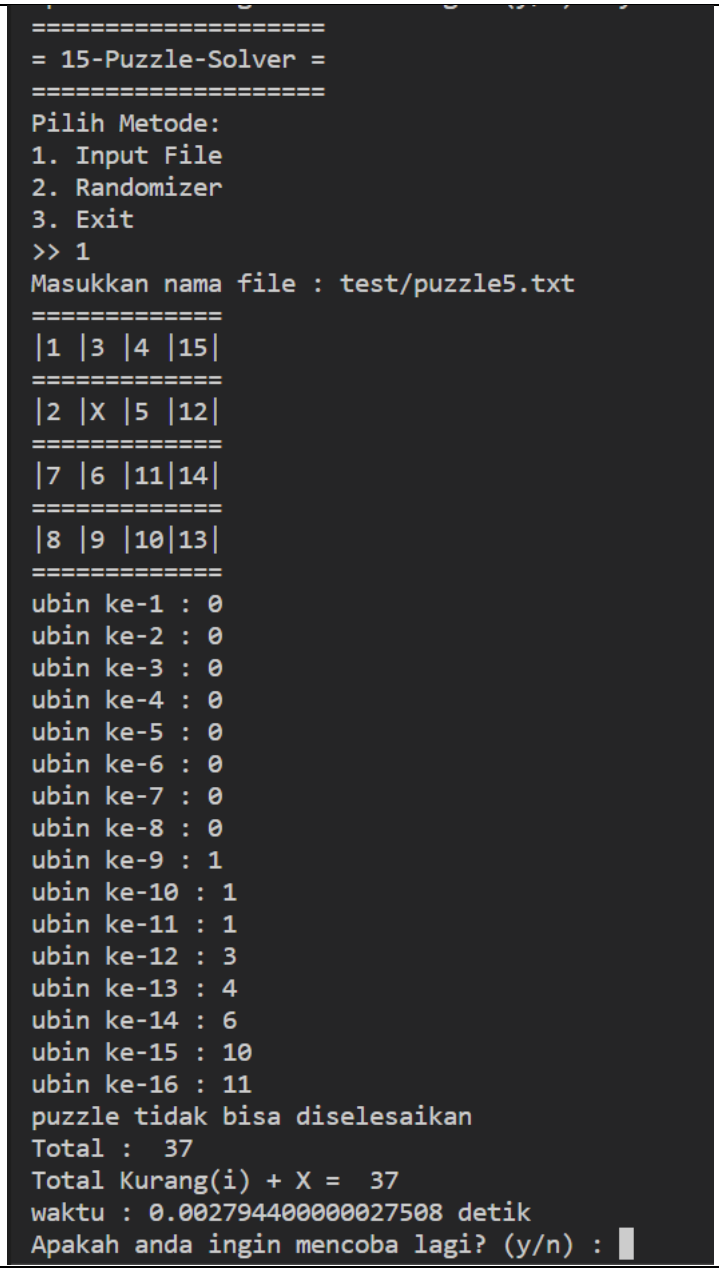
```
=============
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|15|11|
=============
|13|X |14|12|
=============
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|15|11|
=============
|13|14|X |12|
=============
>> Atas
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|X |11|
=============
|13|14|15|12|
=============
>> Kanan
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|X |
=============
|13|14|15|12|
=============
>> Bawah
=============
|1 |2 |3 |4 |
=============
|5 |6 |7 |8 |
=============
|9 |10|11|12|
=============
|13|14|15|X |
=============
Jumlah simpul yang dibangkitkan : 45
waktu : 0.02901800000000776 detik
Apakah anda ingin mencoba lagi? (y/n) :
```

4. Uji Test 4

Puzzle4.txt

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

```
====================
= 15-Puzzle-Solver =
====================
Pilih Metode:
1. Input File
2. Randomizer
3. Exit
>> 1
Masukkan nama file : test/puzzle4.txt
=============
|X |3 |15|14|
=============
|1 |4 |2 |11|
=============
|7 |5 |12|6 |
=============
|8 |9 |10|13|
=============
ubin ke-1 : 0
ubin ke-2 : 0
ubin ke-3 : 0
ubin ke-4 : 0
ubin ke-5 : 0
ubin ke-6 : 0
ubin ke-7 : 0
ubin ke-8 : 0
ubin ke-9 : 1
ubin ke-10 : 2
ubin ke-11 : 2
ubin ke-12 : 4
ubin ke-13 : 6
ubin ke-14 : 12
ubin ke-15 : 13
ubin ke-16 : 15
puzzle tidak bisa diselesaikan
Total :  55
Total Kurang(i) + X =  55
waktu : 0.0032544000000029882 detik
Apakah anda ingin mencoba lagi? (y/n) :
```

5. Uji Test 5 Puzzle5.txt 1 3 4 15 2 16 5 12 7 6 11 14 8 9 10 13	

C. Checklist

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		✓

D. Kode Program

a. Solver.py

```
import copy

class Solver:
    def __init__(self):
        self.matriks = [[0 for i in range(4)] for i in range(4)]
        self.queue = []
        self.langkah = ""
        self.depth = 0
        self.kurang = []
        self.simpul = {}

    # fungsi untuk menampilkan Puzzle
    def printMatriks(self):
        print("=============")
        for i in range(4):
            for j in range(4):
                print("|",end="")
                if (self.matriks[i][j] != 16):
                    print(self.matriks[i][j], end="")
                    if(self.matriks[i][j] < 10):
                        print(" ", end="")
                else:
                    print("X ", end="")
            print("|")
            if(i != 3):
                print("=============")
        print("=============")

    # fungsi untuk mengubah dari bentuk matriks (per-empat bagian)
    # ke bentuk list (satu bagian saja)
    def convert(self,matriks):
        matrix = [0 for i in range(16)]
        a = 0
        for i in range(4):
            for j in range(4):
                matrix[a] = matriks[i][j]
                a += 1
        return matrix

    # fungsi untuk mencari ubin yang kosong
    def cariUbinKosong(self):
        i = 0
        ketemu = False
        while(not ketemu and i < 4):
            j = 0
```

```
            while(not ketemu and j < 4):
                if(self.matriks[i][j] == 16):
                    ketemu = True
                j = j + 1
            i = i + 1
        return [i-1,j-1]

    #fungsi untuk mendapatkan nilai X
    def cariX(self):
        i,j = self.cariUbinKosong()[0], self.cariUbinKosong()[1]
        hasil = (i+j+2) % 2
        if(hasil == 0):
            return 0
        else:
            return 1

    # fungsi untuk menghitung nilai cost
    def cost(self,matriks):
        total = 0
        count = 1
        for i in range(4):
            for j in range(4):
                if(matriks[i][j] == count):
                    total += 1
                count += 1
        return total

    # fungsi untuk mengecek apakah puzzle bisa diselesaikan atau tidak
    def isSolve(self):
        hasil = self.hitungInversi() + self.cariX()
        if(hasil % 2 != 0):
            return False
        else:
            return True

    #fungsi untuk mengecek apakah sudah dalam bentuk akhir
    def isFinal(self):
        if(self.cost(self.matriks) != 16):
            return False
        else:
            return True

    #fungsi untuk menghitung nilai "Kurang" dari puzzle
    def hitungInversi(self):
        matriks = []
        for i in range(4):
            for j in range(4):
                matriks.append(self.matriks[i][j])
```

```
        total = 0
        for i in range(16):
            inversi = 0
            for j in range(i+1,16):
                if(matriks[j] < matriks[i]):
                    total += 1
                    inversi += 1
            self.kurang.append([matriks[i],inversi])
        inversi = self.kurang
        inversi.sort(key=lambda x: x[0])
        return total

    # fungsi untuk menampilkan nilai kurang pada setiap ubin
    def printInversi(self):
        for i in range (16):
            print("ubin ke-" + str(i+1) + " : " + str(self.kurang[i][1]))

    # fungsi untuk mengecek setiap pergerakan ubin
    def validasiGerak(self, matriks):
        x = self.cariUbinKosong()[0]
        y = self.cariUbinKosong()[1]
        if(matriks[1] == "bawah"):
            if(x+1 == 4):
                return False
        elif(matriks[1] == "atas"):
            if(x+1 == 1):
                return False
        elif(matriks[1] == "kanan"):
            if(y+1 == 4):
                return False
        elif(matriks[1] == "kiri"):
            if(y+1 == 1):
                return False
        return True

    #fungsi untuk menggerakkan ubin
    def gerak(self,matriks):
        x = self.cariUbinKosong()[0]
        y = self.cariUbinKosong()[1]
        i,j = x, y
        if(self.validasiGerak(matriks)):
            if(matriks[1] == "bawah"):
                self.swapper(matriks, 0, i, j, i+1, j)
            elif(matriks[1] == "atas"):
                self.swapper(matriks, 0, i, j, i-1, j)
            elif(matriks[1] == "kanan"):
                self.swapper(matriks, 0, i, j, i, j+1)
            if(matriks[1] == "kiri"):
```

```
                self.swapper(matriks, 0, i, j, i, j-1)
        return matriks

    # fungsi untuk swap nilai pada matriks karena adanya pergerakan
    def swapper(self,matriks, a, i, j, x, y):
        matriks[a][i][j] = matriks[a][x][y]
        matriks[a][x][y] = 16

    #fungsi untuk menampilkan keterangan navigasi pergerakan
    def navigasi(self):
        now = self.langkah.pop(0)
        if (now == 'keatas') :
            self.gerak([self.matriks, 'atas'])
            print(">> Atas")
        elif (now == 'kebawah') :
            self.gerak([self.matriks, 'bawah'])
            print(">> Bawah")
        elif (now == 'kekiri') :
            self.gerak([self.matriks, 'kiri'])
            print(">> Kiri")
        elif (now == 'kekanan') :
            self.gerak([self.matriks, 'kanan'])
            print(">> Kanan")
        else :
            pass

    #fungsi untuk mengalokasikan queue
    def alokasiQueue(self, posisi, gerak, command, arah):
        posisi = 16 - self.cost(gerak[0]) + self.depth
        self.simpul[tuple(self.convert(gerak[0]))] = command
        self.queue.append([posisi,self.depth,self.langkah + arah,
self.convert(gerak[0])])

    #fungsi untuk menyalin matriks pada setiap arah
    def copyMatriks(self,matriks,command):
        posisi = []
        if(command == 'atas'):
            posisi = [copy.deepcopy(matriks), command]
        if(command == 'bawah'):
            posisi = [copy.deepcopy(matriks), command]
        if(command == 'kiri'):
            posisi = [copy.deepcopy(matriks), command]
        if(command == 'kanan'):
            posisi = [copy.deepcopy(matriks), command]
        return posisi
```

```
 #algoritma Branch and Bound
def BnB(self):
    posUp = 0
    atas = self.copyMatriks(self.matriks,'atas')
    self.gerak(atas)
    posbawah = 0
    bawah = self.copyMatriks(self.matriks,'bawah')
    self.gerak(bawah)
    poskiri = 0
    kiri = self.copyMatriks(self.matriks,'kiri')
    self.gerak(kiri)
    poskanan = 0
    kanan = self.copyMatriks(self.matriks,'kanan')
    self.gerak(kanan)
    self.depth += 1
    if(tuple(self.convert(bawah[0])) not in self.simpul):
        self.alokasiQueue(posbawah, bawah, 'bawah', 'kebawah ')
    if(tuple(self.convert(atas[0])) not in self.simpul):
        self.alokasiQueue(posUp, atas, 'atas', 'keatas ')
    if(tuple(self.convert(kanan[0])) not in self.simpul):
        self.alokasiQueue(poskanan, kanan, 'kanan', 'kekanan ')
    if(tuple(self.convert(kiri[0])) not in self.simpul):
        self.alokasiQueue(poskiri, kiri, 'kiri', 'kekiri ')
    self.queue.sort()
    pop = self.queue.pop(0)
    self.depth,self.langkah = pop[1],pop[2]
    newMatriks = [[0 for i in range (4)] for i in range(4)]
    a = 0
    for i in range(4):
        for j in range(4):
            newMatriks[i][j] = pop[3][a]
            a+=1
    self.matriks = newMatriks
```

b. File.py

```
from Solver import *
from random import shuffle
import numpy as np
class File:
    def __init__(self):
        pass

    # fungsi untuk mengubah file txt menjadi matriks
    def txtToMatriks(self, filename):
        matrix = []
        with open(filename) as f:
            for item in f:
                matrix.append([int(i) for i in item.split()])
        puzzle = Solver()
        puzzle.matriks = matrix
        return puzzle

    # fungsi untuk melakukan pengacakan matriks
    def randomizer(self):
        puzzle = [i+1 for i in range(16)]
        shuffle(puzzle)
        matriks = np.reshape(puzzle, (4,4)).astype('int32')
        Puzzle = Solver()
        Puzzle.matriks = matriks
        return Puzzle
```

c. Main.py

```
from File import *
from Solver import *
from time import perf_counter
x = True
while(x):
    print("====================")
    print("= 15-Puzzle-Solver =")
    print("====================")
    print("Pilih Metode: ")
    print("1. Input File")
    print("2. Randomizer")
    print("3. Exit")
    inputPilihan = int(input(">> "))

   if(inputPilihan == 1):
        print("contoh: test/puzzlex.txt")
        inputfile = input("Masukkan nama file : ")
        file = File()
        puzzle = file.txtToMatriks(inputfile)
        puzzle.printMatriks()
        waktuMulai = perf_counter()
        if (puzzle.isSolve()) :
            puzzle.printInversi()
            hasil = puzzle.hitungInversi()
            print("Nilai fungsi kurang untuk tiap ubin: ")
            print("Total : ", hasil)
            print("Total Kurang(i) + X = ", hasil + puzzle.cariX())
            print("Status: ")
            print("Puzzle bisa diselesaikan")
            temp = copy.deepcopy(puzzle)
            puzzle.simpul[tuple(puzzle.convert(puzzle.matriks))] = 'none'
            while (not puzzle.isFinal()) :
                puzzle.BnB()
            temp.langkah = puzzle.langkah.split(" ")
            while (len(temp.langkah) != 1) :
                temp.navigasi()
                temp.printMatriks()
            print("Jumlah simpul yang dibangkitkan : " + str(len(puzzle.simpul) - 1))
        else :
            puzzle.printInversi()
            print("puzzle tidak bisa diselesaikan")
            hasil = puzzle.hitungInversi()
            print("Total : ", hasil)
            print("Total Kurang(i) + X = ", hasil + puzzle.cariX())
        waktuSelesai = perf_counter()
```

```
        print("waktu : " + str(waktuSelesai-waktuMulai) + " detik")

    elif(inputPilihan == 2):
        file = File()
        puzzle = file.randomizer()
        puzzle.printMatriks()
        waktuMulai = perf_counter()
        if (puzzle.isSolve()) :
            print("Nilai fungsi kurang untuk tiap ubin: ")
            puzzle.printInversi()
            hasil = puzzle.hitungInversi()
            print("Total : ", hasil)
            print("Total Kurang(i) + X = ", hasil + puzzle.cariX())
            print("Puzzle bisa diselesaikan")
            temp = copy.deepcopy(puzzle)
            puzzle.simpul[tuple(puzzle.convert(puzzle.matriks))] = 'none'
            while (not puzzle.isFinal()) :
                puzzle.BnB()
            temp.langkah = puzzle.langkah.split(" ")
            while (len(temp.langkah) != 1) :
                temp.navigasi()
                temp.printMatriks()
            print("Jumlah simpul yang dibangkitkan : " + str(len(puzzle.simpul) - 1))
        else :
            puzzle.printInversi()
            print("puzzle tidak bisa diselesaikan")
            hasil = puzzle.hitungInversi()
            print("Total : ", hasil)
            print("Total Kurang(i) + X = ", hasil + puzzle.cariX())
        waktuSelesai = perf_counter()
        print("waktu : " + str(waktuSelesai-waktuMulai) + " detik")

    elif(inputPilihan == 3):
        exit()

    endstate = input("Apakah anda ingin mencoba lagi? (y/n) : ")
    if(endstate == "n"):
        print("\n")
        print("==================================")
        print("==Terima Kasih sudah bermain !!!==")
        print("==================================")
        print("\n")
        x = False
```

E. Berkas teks berisi contoh instansi 5 buah persoalan 15-puzzle

1. Puzzle1.txt

1 2 3 4 5 6 8 11 9 10 7 16

```
13 14 15 12
```

2. Puzzle2.txt

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

3. Puzzle3.txt

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

4. Puzzle4.txt

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

5. Puzzle5.txt

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

F. Link

Link Github: https://github.com/kibare/15-Puzzle-Solver.git