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

Dibuat oleh:

Aditya Prawira Nugroho 13520049

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

A. Algoritma

1. Branch and Bound

Algoritma Branch and Bound adalah algoritma pencarian solusi yang digunakan untuk persoalan optimisasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan (*constraints*) persoalan yang sudah didefinisikan. Pada dasarnya, algoritma Branch and Bound adalah algoritma didasari oleh BFS + least cost search. Algoritma Branch and Bound memiliki dua karakteristik. Pertama, setiap simpul akan diberi sebuah nilai *cost* ĉ(i) —nilai taksiran linstasan termurah ke simpul status tujuan yang melalui simpul status i. Kedua, simpul yang akan di-*expand* berikutnya tidak lagi urut berdasarkan pembangkitannya, tetapi urut dari simpul dengan *cost* yang paling kecil untuk kasus minimasi dan yang paling besar untuk kasus maksimasi.

2. Penyelesaian 15-Puzzle Menggunakan Algoritma Branch and Bound

Pertama, saya membuat sebuah objek FifteenPuzzle terlebih dahulu. Objek ini memiliki atribut dua buah array of integer dengan panjang 16, satu array untuk menyimpan state puzzle dan satu array untuk menyimpan KURANG(i), yaitu banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i), dari state puzzle dan nullPosition, yaitu atribut yang menyimpan indeks ubin kosong. Dalam program saya, ubin kosong dilambangkan dengan angka 0.

Program akan membaca sebuah input dari file .txt atau randomizer. Setelah itu, data state puzzle akan digunakan untuk membuat objek FifteenPuzzle yang diberi nama root. Sebelum puzzle diproses, dicek terlebih dahulu apakah puzzle yang diinput dapat diselesaikan dengan menjumlahkan KURANG(i) + X, dengan X adalah posisi ubin kosong. X akan bernilai 1 jika jumlah dari indeks baris dan kolom ubin kosong bernilai ganjil, bernilai 0 jika genap. Jika hasil penjumlahan KURANG(i) dengan X bernilai ganjil, maka puzzle tidak dapat diselesaikan dan program berhenti. Sebaliknya, program akan memproses puzzle jika penjumlahan tersebut bernilai genap.

Program akan mengecek gerakan yang memungkinkan untuk puzzle. Gerakan yang dapat dilakukan ada 4, yaitu atas, bawah, kiri, dan kanan. Setelah program menemukan gerakan yang mungkin, anak dari simpul dibangkitkan sesuai gerakan yang mungkin. Kemudian, dihitung *cost* dari simpul ke-i, yaitu kedalaman simpul ditambah ongkos mencapai simpul tujuan dari simpul ke-i. Setelah itu, anak dari root dimasukkan ke dalam sebuah Priority Queue dengan *cost* simpul ke-i sebagai prioritas. Simpul selanjutnya yang diekspan adalah kepala dari Priority Queue.

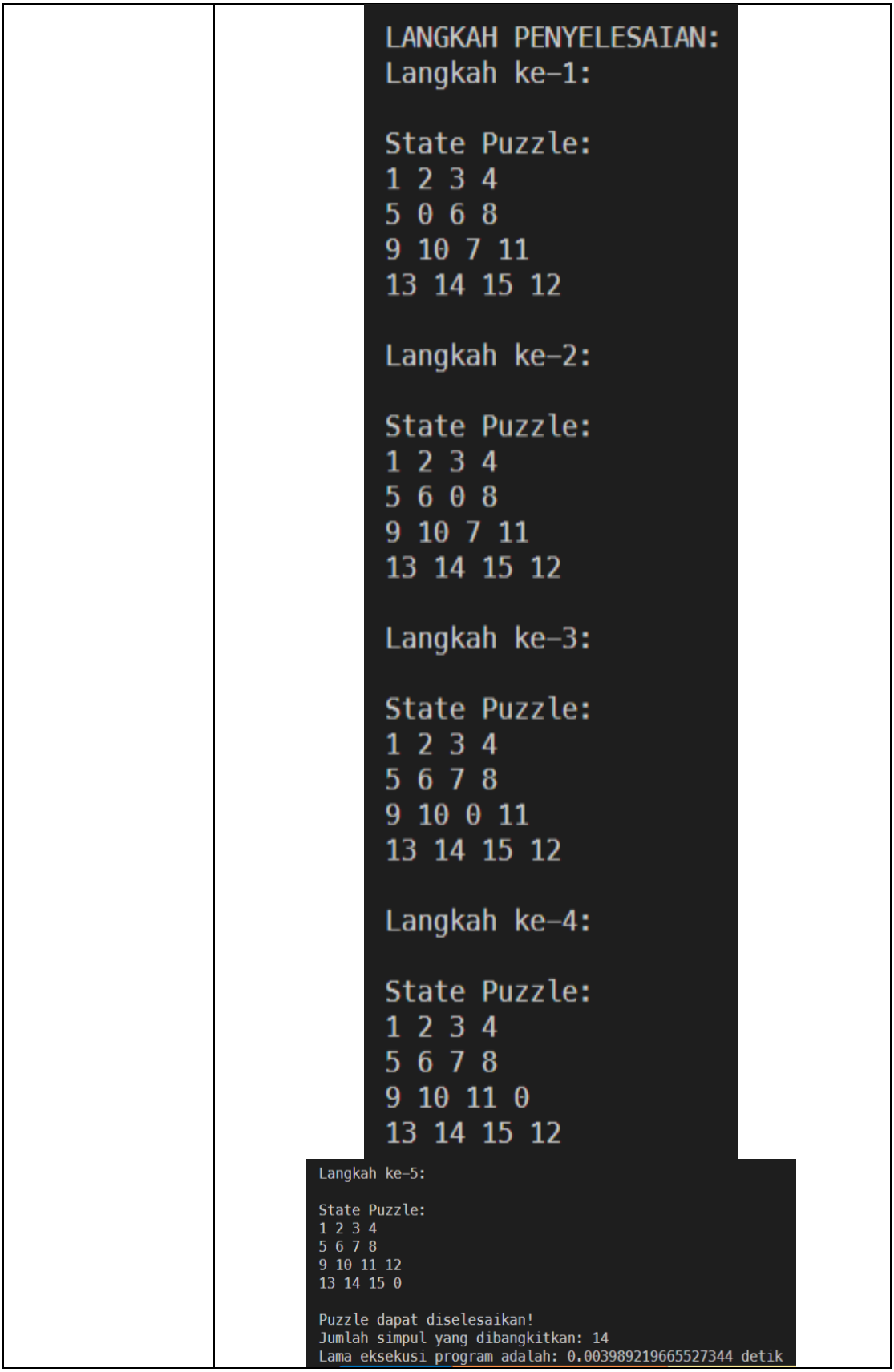
Setelah didapatkan simpul ekspan selanjutnya, dicek terlebih dahulu apakah simpul yang sedang diekspan merupakan solusi atau bukan. Jika iya, program berhenti dan mencatat gerakan yang sudah dilakukan, kemudian mengeluarkan langkah-langkah penyelesaian puzzle. Jika bukan solusi, maka program akan mengecek apakah simpul sudah pernah diekspan sebelumnya, jika sudah pernah maka program akan

mengekspan simpul baru yang ada di dalam Priority Queue. Langkah-langkah tersebut akan diulangi hingga ditemukan solusi.

B. Screenshot Input Output Program

1. Kasus puzzle1.txt

Input		~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~ Anda ingin input puzzle darimana? 1. Input puzzle dari file 2. Input puzzle secara acak Masukkan angka pilihan Anda: > 1 Masukkan nama file Anda (tanpa .txt): > puzzle1
Output		State Puzzle: 1 2 3 4 0 5 6 8 9 10 7 11 13 14 15 12 KURANG(i): 1 : 0 2 : 0 3 : 0 4 : 0 5 : 0 6 : 0 7 : 0 8 : 1 9 : 1 10 : 1 11 : 0 12 : 0 13 : 1 14 : 1 15 : 1 Total KURANG(i) + X: 18


LANGKAH PENYELESAIAN:
Langkah ke-1:
State Puzzle:
1 2 3 4
5 0 6 8
9 10 7 11
13 14 15 12
Langkah ke-2:
State Puzzle:
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12
Langkah ke-3:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12
Langkah ke-4:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 11 0
13 14 15 12
Langkah ke-5:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
Puzzle dapat diselesaikan!
Jumlah simpul yang dibangkitkan: 14
Lama eksekusi program adalah: 0.003989219665527344 detik

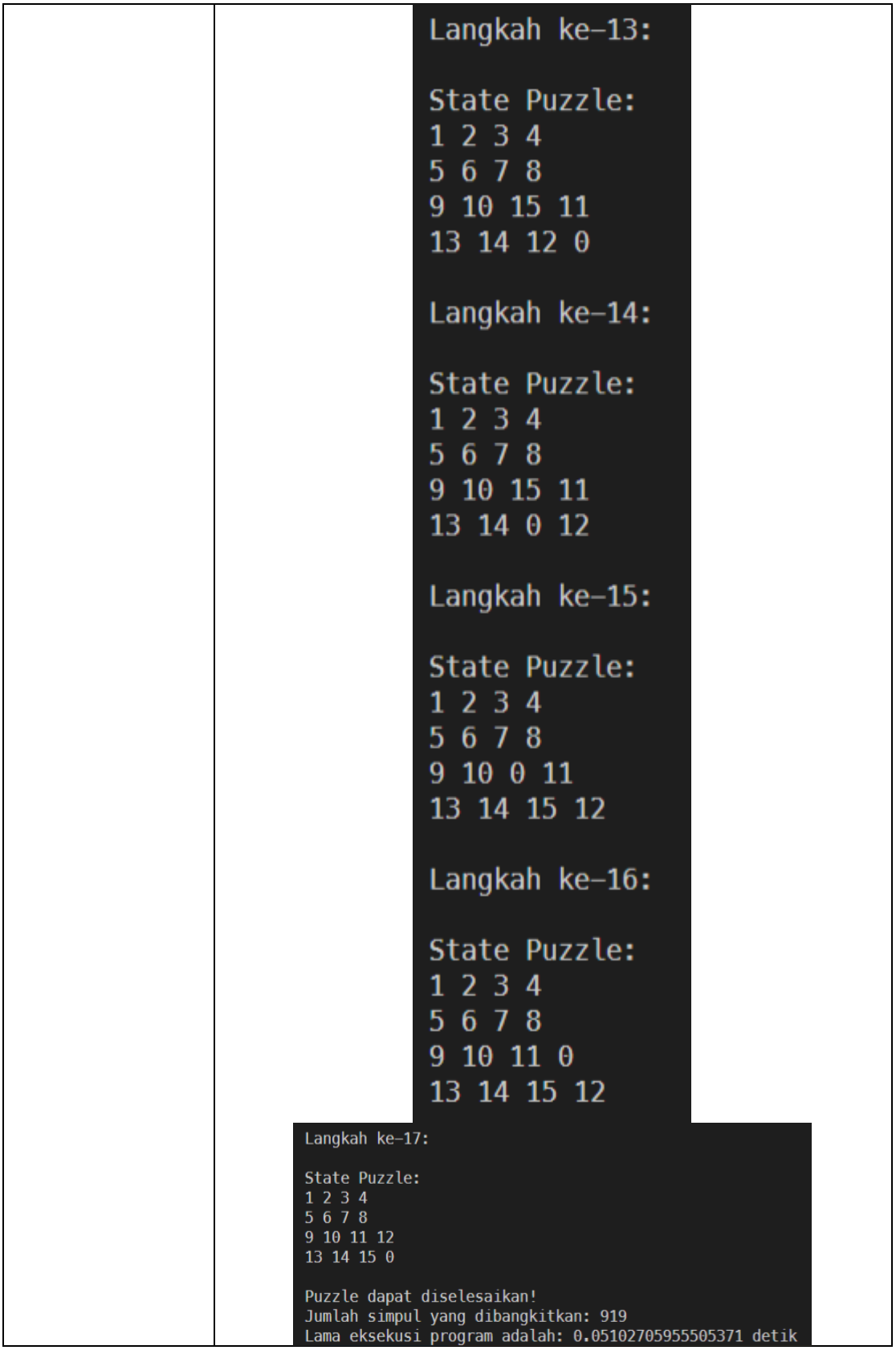
2. Kasus puzzle2.txt

Input		~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~ Anda ingin input puzzle darimana? 1. Input puzzle dari file 2. Input puzzle secara acak Masukkan angka pilihan Anda: > 1 Masukkan nama file Anda (tanpa .txt): > puzzle2
Output		State Puzzle: 6 5 2 4 9 1 3 8 10 0 7 15 13 14 12 11 KURANG(i): 1 : 0 2 : 1 3 : 0 4 : 2 5 : 4 6 : 5 7 : 0 8 : 1 9 : 4 10 : 1 11 : 0 12 : 1 13 : 2 14 : 2 15 : 4 Total KURANG(i) + X: 34

```
LANGKAH PENYELESAIAN:
Langkah ke-1:

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

Langkah ke-2:

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

Langkah ke-3:

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

Langkah ke-4:

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

		Langkah ke-5: State Puzzle: 0 6 2 4 1 5 3 8 9 10 7 15 13 14 12 11 Langkah ke-6: State Puzzle: 1 6 2 4 0 5 3 8 9 10 7 15 13 14 12 11 Langkah ke-7: State Puzzle: 1 6 2 4 5 0 3 8 9 10 7 15 13 14 12 11 Langkah ke-8: State Puzzle: 1 0 2 4 5 6 3 8 9 10 7 15 13 14 12 11	

```
Langkah ke-9:

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

Langkah ke-10:

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

Langkah ke-11:

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

Langkah ke-12:

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


Langkah ke-13:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 15 11
13 14 12 0
Langkah ke-14:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 15 11
13 14 0 12
Langkah ke-15:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 0 11
13 14 15 12
Langkah ke-16:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 11 0
13 14 15 12
Langkah ke-17:
State Puzzle:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
Puzzle dapat diselesaikan!
Jumlah simpul yang dibangkitkan: 919
Lama eksekusi program adalah: 0.05102705955505371 detik

3. Kasus puzzle3.txt

Input		~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~ Anda ingin input puzzle darimana? 1. Input puzzle dari file 2. Input puzzle secara acak Masukkan angka pilihan Anda: > 1 Masukkan nama file Anda (tanpa .txt): > puzzle3
Output		State Puzzle: 1 2 3 4 5 6 7 8 9 12 15 14 13 11 0 10 KURANG(i): 1 : 0 2 : 0 3 : 0 4 : 0 5 : 0 6 : 0 7 : 0 8 : 0 9 : 0 10 : 0 11 : 1 12 : 2 13 : 2 14 : 3 15 : 4 Total KURANG(i) + X: 14

```
Langkah ke-1:

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

Langkah ke-2:

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

Langkah ke-3:

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

Langkah ke-4:

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

Langkah ke-5:

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

```
Langkah ke-6:

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

Langkah ke-7:

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

Langkah ke-8:

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

Langkah ke-9:

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

Langkah ke-10:

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

		Langkah ke-11: State Puzzle: 1 2 3 4 5 6 7 8 9 10 11 12 13 0 14 15 Langkah ke-12: State Puzzle: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0 15 Langkah ke-13: State Puzzle: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 Puzzle dapat diselesaikan! Jumlah simpul yang dibangkitkan: 398 Lama eksekusi program adalah: 0.023993492126464844 detik

4. Kasus puzzle4.txt

Input		~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~ Anda ingin input puzzle darimana? 1. Input puzzle dari file 2. Input puzzle secara acak Masukkan angka pilihan Anda: > 1 Masukkan nama file Anda (tanpa .txt): > puzzle4
Output		State Puzzle: 3 1 2 4 0 5 7 8 10 6 15 12 9 13 14 11 KURANG(i): 1 : 0 2 : 0 3 : 2 4 : 0 5 : 0 6 : 0 7 : 1 8 : 1 9 : 0 10 : 2 11 : 0 12 : 2 13 : 1 14 : 1 15 : 5 Total KURANG(i) + X: 27 FIFTEEN PUZZLE TIDAK DAPAT DISELESAIKAN Lama eksekusi program adalah: 0.003993511199951172 detik

5. Kasus puzzle5.txt

Input	~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~ Anda ingin input puzzle darimana? 1. Input puzzle dari file 2. Input puzzle secara acak Masukkan angka pilihan Anda: > 1 Masukkan nama file Anda (tanpa .txt): > puzzle5
Output	State Puzzle: 15 4 10 1 9 14 5 8 12 2 7 6 11 3 13 0 KURANG(i): 1 : 0 2 : 0 3 : 0 4 : 3 5 : 2 6 : 1 7 : 2 8 : 4 9 : 6 10 : 8 11 : 1 12 : 5 13 : 0 14 : 9 15 : 14 Total KURANG(i) + X: 55 FIFTEEN PUZZLE TIDAK DAPAT DISELESAIKAN Lama eksekusi program adalah: 0.00400090217590332 detik

C. Source Program

1. FifteenPuzzle.py

```
# buat array sepanjang 16, 1 nya null
class FifteenPuzzle:
  # Posisi kosong ditandai dengan angka 0
  puzzle = [0 for i in range(16)]
  kurang = [0 for i in range(16)]
  nullPosition = 0

  # Konstruktor yang menerima puzzlenya seperti apa, kemudian mencari
posisi 0 dan menginisiasi array kurang
  def __init__(self, puzzle):
    self.puzzle = puzzle
    self.nullPosition = puzzle.index(0)
    for i in range(16):
      self.hitungKurang(i)

  # Untuk print puzzle
  def __str__(self):
    print("\nState Puzzle:")
    for i in range(16):
```

```
      print(self.puzzle[i], end=" ")
      if (i+1) % 4 == 0:
        print("")
    return ""

  def printInfo(self):
    print("KURANG(i):")
    for i in range(1, len(self.kurang)):
      print(f"{i}\t: {self.kurang[i]}")
    print(f"Total KURANG(i) + X: {self.totalKurang()}\n")

  # Untuk mengecek apakah ubin kosong ada di indeks ganjil atau genap,
return 1 jika ganjil
  def cekUbinKosong(self):
    # Jika ubin kosong ada di indeks ganjil, maka return 1
    if self.nullPosition % 2 == 1:
      if 0 <= self.nullPosition <= 3 or 8 <= self.nullPosition <= 11:
        return 1
      else:
        return 0
    else:
      if 4 <= self.nullPosition <= 7 or 12 <= self.nullPosition <= 15:
        return 1
      else:
        return 0

  # Fungsi mereturn jumlah kurang(i) + x
  def totalKurang(self):
    sum = 0
    for i in range(16):
      sum += self.kurang[i]
    sum += self.cekUbinKosong()
    return sum

  # Untuk mengecek apakah puzzle dapat diselesaikan
  def isPuzzleSolvable(self):
    sum = self.totalKurang()
    if sum % 2 == 0:
      return True
    else:
      return False

  # Menghitung kurang(Number) dari puzzle
  def hitungKurang(self, Number):
    if Number == 0:
      self.kurang[0] = 15 - self.nullPosition
    else:
      # Buat status apakah Number telah ditemukan di puzzle
```

```
      foundNumber = False
      for i in range(16):
        # Ubah status jika Number telah ditemukan
        if self.puzzle[i] == Number:
          foundNumber = True
        # Jika status = True, cek apakah puzzle[i] < Number, jika iya,
kurang[idxKurang] += 1
        if foundNumber:
          if self.puzzle[i] < Number and self.puzzle[i] != 0:
            self.kurang[Number] += 1

  # Menghitung ada berapa ubin yang tidak sesuai tempat jika dibandingkan
dengan goal state (g(node) kalo di ppt)
  def g(self):
    sum = 0
    for i in range(16):
      if self.puzzle[i] != 0:
        if self.puzzle[i] != i + 1:
          sum += 1
    return sum

  # Method untuk menggerakkan up, down, left, right
  def moveUp(self):
    if 0 <= self.nullPosition <= 3:
      print("Tidak bisa gerak ke atas")
    else:
      self.puzzle[self.nullPosition], self.puzzle[self.nullPosition - 4] =
self.puzzle[self.nullPosition - 4], self.puzzle[self.nullPosition]
      self.nullPosition -= 4
      for i in range(16):
        self.hitungKurang(i)

  def moveDown(self):
    if 12 <= self.nullPosition <= 15:
      print("Tidak bisa gerak ke bawah")
    else:
      self.puzzle[self.nullPosition], self.puzzle[self.nullPosition + 4] =
self.puzzle[self.nullPosition + 4], self.puzzle[self.nullPosition]
      self.nullPosition += 4
      for i in range(16):
        self.hitungKurang(i)

  def moveLeft(self):
    if self.nullPosition % 4 == 0:
      print("Tidak bisa gerak ke kiri")
    else:
      self.puzzle[self.nullPosition], self.puzzle[self.nullPosition - 1] =
self.puzzle[self.nullPosition - 1], self.puzzle[self.nullPosition]
```

```
      self.nullPosition -= 1
      for i in range(16):
        self.hitungKurang(i)

  def moveRight(self):
    if self.nullPosition % 4 == 3:
      print("Tidak bisa gerak ke kanan")
    else:
      self.puzzle[self.nullPosition], self.puzzle[self.nullPosition + 1] =
self.puzzle[self.nullPosition + 1], self.puzzle[self.nullPosition]
      self.nullPosition += 1
      for i in range(16):
        self.hitungKurang(i)

  # Method untuk menggerakkan puzzle sesuai enum move
  def move(self, move):
    if move == 0:
      self.moveUp()
    elif move == 1:
      self.moveDown()
    elif move == 2:
      self.moveLeft()
    elif move == 3:
      self.moveRight()

  # Method untuk menggerakkan puzzle tetapi berlawanan dengan enum move
  def moveOpposite(self, move):
    if move == 0:
      self.moveDown()
    elif move == 1:
      self.moveUp()
    elif move == 2:
      self.moveRight()
    elif move == 3:
      self.moveLeft()

  # Method untuk mendapatkan gerak puzzle ke atas, bawah, kiri, atau
kanan. 0 untuk atas, 1 untuk bawah
  # 2 untuk kiri, 3 untuk kanan
  def getLegalMove(self):
    # Return array dari move yang legal, dalam bentuk enumerasi
    cond1 = 0 <= self.nullPosition <= 3
    cond2 = 12 <= self.nullPosition <= 15
    cond3 = self.nullPosition % 4 == 0
    cond4 = self.nullPosition % 4 == 3
    if cond1:
      if cond3:
        return [1, 3]
```

```
      elif cond4:
        return [1, 2]
      else:
        return [1, 2, 3]
    elif cond2:
      if cond3:
        return [0, 3]
      elif cond4:
        return [0, 2]
      else:
        return [0, 2, 3]
    else:
      if cond3:
        return [0, 1, 3]
      elif cond4:
        return [0, 1, 2]
      else:
        return [0, 1, 2, 3]

  # Method untuk mengecek apakah kedua puzzle sama
  def isEqualtoPuzzle(self, puzzle):
    return self.puzzle == puzzle.puzzle

  def isEqualtoArray(self, array):
    return self.puzzle == array

  def getOppositeMove(self, move):
    if move == 0:
      return 1
    elif move == 1:
      return 0
    elif move == 2:
      return 3
    elif move == 3:
      return 2
```

2. util.py

```
import random

def txtToArray(fileName):
    # Reads a text file and returns an array of number
    # Inisiasi array
    arrayOfNum = []
    with open(fileName, 'r') as f:
        lines = f.readlines()
        for line in lines:
            # Pisah string berdasarkan spasi
```

```
            line = line.split(" ")
            for char in line:
                # Append ke array dalam bentuk integer
                if char != '\n' and char != ' ':
                    arrayOfNum.append(int(char))
    return arrayOfNum

def randPuzzle():
    # Returns a random puzzle
    puzzle = []
    while len(puzzle) < 16:
        # Generate angka random 0 hingga 15
        num = random.randint(0,15)
        # Jika angka belum ada di puzzle, append
        if num not in puzzle:
            puzzle.append(num)
    return puzzle
```

3. Main.py

```
import FifteenPuzzle as fp
from util import *
from queue import PriorityQueue
import os
import time
from os.path import exists

# Tanya pengguna ingin input puzzle darimana
print("~~~SELAMAT DATANG DI SOLVER 15PUZZLE~~~")
print("Anda ingin input puzzle darimana?")
print("1. Input puzzle dari file")
print("2. Input puzzle secara acak")
genPuzzle = input("Masukkan angka pilihan Anda: \n> ")
while genPuzzle != "1" and genPuzzle != "2":
    print("Pilihan tidak tersedia, silahkan masukkan angka 1 atau 2")
    genPuzzle = input("Masukkan angka pilihan Anda: \n> ")

# Jika pengguna memilih 1
if genPuzzle == "1":
    fileName = input("Masukkan nama file Anda (tanpa .txt): \n> ")
    dirAwal = os.getcwd()
    arrOfDir = dirAwal.split("\\")
    if arrOfDir[-1] == "src":
        dirAwal = os.path.dirname(dirAwal)
    dirAwal += "\\test\\"
    while not os.path.exists(dirAwal + fileName + ".txt"):
        print("File tidak ditemukan, silahkan masukkan nama file yang
benar")
        fileName = input("Masukkan nama file Anda (tanpa .txt): \n> ")
```

```
    stateAwal = txtToArray(dirAwal + fileName + ".txt")
else:
    stateAwal = randPuzzle()

# Buat objek puzzle
root = fp.FifteenPuzzle(stateAwal)

# Inisiasi variabel
# Kedalaman tree didapat dari length list of prev moves
listOfSimpulEkspan = {}
found = False
solution = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0]
jumlahSimpulYangDibangkitkan = 0
unique = 0

# Mulai waktu dari sini
startTime = time.time()

# Cek apakah puzzle solvable
if not root.isPuzzleSolvable():
    print(root)
    root.printInfo()
    endTime = time.time()
    print("FIFTEEN PUZZLE TIDAK DAPAT DISELESAIKAN")
    print("Lama eksekusi program adalah: " + str(endTime - startTime) + "
detik")
else:
    # Buat prioqueue simpul hidup
    # Isi dari queue adalah (priority, [puzzle, [list of prev moves]])
    simpulHidup = PriorityQueue()
    print(root)
    root.printInfo()
    # Insert akar ke listOfSimpulEkspan dan simpul hidup
    simpulHidup.put((1, unique, [root, []]))
    listOfSimpulEkspan[(*root.puzzle,)] = True

    # Cek apakah root sudah solusi
    if root.isEqualtoArray(solution):
        endTime = time.time()
        print(root)
        print(f"Jumlah simpul yang dibangkitkan:
{jumlahSimpulYangDibangkitkan}")
        print("Lama eksekusi program adalah: " + str(endTime - startTime)
+ " detik")
    else:
        simpulEkspan = simpulHidup.get()
        # tambahkan simpul hidup yang mungkin dari root
        legalMoves = simpulEkspan[2][0].getLegalMove()
```

```
        for move in legalMoves:
            # Buat simpul baru
            simpulBaru =
fp.FifteenPuzzle(simpulEkspan[2][0].puzzle.copy())
            simpulBaru.move(move)
            # Masukkan simpul baru ke simpul hidup
            cost = len(simpulEkspan[2][1]) + 1 + simpulBaru.g()
            listOfPrevMoves = simpulEkspan[2][1] + [move]
            simpulHidup.put((cost, unique - 1, [simpulBaru,
listOfPrevMoves]))
            unique -= 1
            jumlahSimpulYangDibangkitkan += 1
            listOfSimpulEkspan[(*simpulBaru.puzzle,)] = True
        while not found and not simpulHidup.empty():
            simpulEkspan = simpulHidup.get()
            if simpulEkspan[2][0].isEqualtoArray(solution):
                found = True
                endTime = time.time()
                # print langkah langkah simpul
                print("LANGKAH PENYELESAIAN:")
                i = 1
                for move in simpulEkspan[2][1]:
                    print("Langkah ke-" + str(i) + ":")
                    root.move(move)
                    print(root)
                    i += 1
                print("Puzzle dapat diselesaikan!")
                print(f"Jumlah simpul yang dibangkitkan:
{jumlahSimpulYangDibangkitkan}")
                print("Lama eksekusi program adalah: " + str(endTime -
startTime) + " detik")
            else:
                # Bangkitkan anaknya
                legalMoves = simpulEkspan[2][0].getLegalMove()
                for move in legalMoves:
                    # Cek apakah move kebalikan dari move sebelumnya
                    if not move ==
simpulEkspan[2][0].getOppositeMove(simpulEkspan[2][1][-1]):
                        # Buat simpul baru
                        simpulBaru =
fp.FifteenPuzzle(simpulEkspan[2][0].puzzle.copy())
                        simpulBaru.move(move)
                        if not (*simpulBaru.puzzle,) in
listOfSimpulEkspan:
                            # Masukkan simpul baru ke simpul hidup
                            cost = len(simpulEkspan[2][1]) + 1 +
simpulBaru.g()
                            listOfPrevMoves = simpulEkspan[2][1] + [move]
```

```
                            simpulHidup.put((cost, unique - 1,
[simpulBaru, listOfPrevMoves]))
                            unique -= 1
                            jumlahSimpulYangDibangkitkan += 1
                            listOfSimpulEkspan[(*simpulBaru.puzzle,)] =
True
4.
```

D. Berkas Teks Intansiasi 5 Buah Persoalan 15-Puzzle

1. puzzle1.txt

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

2. puzzle2.txt

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

3. puzzle3.txt

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

4. puzzle4.txt

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

5. puzzle5.txt

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

E. Alamat Kode Program

https://github.com/Adityapnn811/Tucil3_13520049

F. Tabel 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		√