Laporan Tugas Kecil 3 IF2211 Strategi Algoritma
Semester II Tahun Akademik 2021/2022

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

Disusun oleh:
Ahmad Alfani Handoyo
13520023
K-02

Program Studi S1 Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

A. Algoritma *Branch and Bound* untuk menyelesaikan 15-Puzzle

Algoritma yang digunakan untuk menyelesaikan 15-Puzzle menggunakan pendekatan *branch and bound*. Algoritma *branch and bound* pada umumnya mirip dengan algoritma *breadth-first search* (BFS) namun ditambah dengan ketentuan simpul berikutnya yang dicari/diekspansi adalah simpul dengan *cost* terkecil. *Cost* dari setiap simpul adalah suatu estimasi ongkos termurah lintasan. Karena *cost* adalah estimasi, maka nilainya dihitung secara heuristik.

Dalam perhitungan *cost* simpul 15-Puzzle, digunakan taksiran *cost* sebagai berikut,

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

dengan $\hat{c}(P)$ adalah *cost* untuk simpul P, $f(P)$ panjang lintasan dari simpul akar ke P, dan $\hat{g}(P)$ adalah taksiran panjang lintasan terpendek dari *P* ke simpul solusi dengan akar P yang diestimasikan dengan menghitung jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir.

Implementasi algoritma *branch and bround* dilakukan dengan membuat sebuah pohon keadaan simpul yang direpresentasikan dengan sebuah kelas yang memiliki atribut konfigurasi puzzle sementara, panjang lintasan dari simpul akar, dan urutan langkah-langkah dari simpul akar. Terdapat pula metode pada kelas untuk menghitung *cost* dari konfigurasi puzzle dan mengubah posisi ubin kosong. Ubin kosong direpresentasikan dengan angka 0. Simpul-simpul ini kemudian dimasukkan ke sebuah *priority queue* bawaan dari Python dengan prioritas *cost* terkecil.

Pendekatan algoritma *branch and bound* yang digunakan dalam menyelesaikan masalah 15-Puzzle menggunakan taksiran *cost* seperti pada rumus di atas dan juga menerapkan beberapa teknik heuristik demi mempercepat alur kerja algoritma. Heuristik pertama yang diterapkan adalah setiap pembangkitan simpul baru mempunyai syarat bahwa konfigurasi puzzle tersebut belum pernah dicek sebelumnya. Heuristik kedua adalah dalam pemilihan simpul dengan *cost* terkecil, bila ada dua atau lebih simpul dengan *cost* terkecil maka akan dipilih simpul yang dibangkitkan paling akhir sehingga mempunyai kedalaman yang lebih dalam. Alasan diterapkannya heuristik ini agar pohon pencarian simpul tidak mencabang besar dari simpul-simpul dengan kedalaman yang lebih dangkal yang berkemungkinan membuahkan lebih banyak kemungkinan solusi yang baru. Heuristik terakhir adalah pembangkitan simpul anak-anak mempertimbangkan langkah sebelumnya yang pernah dilakukan sehingga langkah tidak mengembalikan konfigurasi puzzle kembali ke langkah sebelumnya. Misalnya bila pada langkah sebelumnya memindahkan ubin kosong ke atas, maka pada langkah ini tidak boleh memindahkan ubin kosong ke bawah.

Karena penentuan langkah-langkah untuk satu solusi saja membutuhkan waktu yang cukup lama karena ketidakefisienan *cost* dalam menentukan simpul terbaik untuk diekspansikan, maka pencarian solusi berhenti ketika satu solusi telah dihasilkan.

Secara garis besar, langkah algoritma yang diimplementasikan adalah sebagai berikut:

1. Masukkan simpul akar ke antrean *priority queue*. Bila simpul akar adalah solusi (matriks terurut 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ubin kosong), berhenti.
2. Jika *priority queue* kosong, berhenti.
3. Jika *priority queue* tidak kosong, pilih simpul *i* dengan *cost* terkecil. Bila terdapat beberapa simpul dengan *cost* terkecil, pilih simpul yang dibangkitkan paling akhir.
4. Jika simpul *i* adalah solusi, berhenti.
5. Jika simpul *i* bukan simpul solusi, bangkitkan semua anak-anak untuk langkah atas, bawah, kiri, dan kanan dengan syarat heuristik konfigurasi puzzle belum pernah dicek dan langkah tidak kebalikan dari langkah sebelumnya, dan juga letak ubin kosong memperbolehkan dilakukannya langkah tersebut. Bila simpul *i* tidak mempunyai anak, kembali ke langkah 2.
6. Untuk setiap anak *i*, hitung *cost*-nya dan masukkan anak-anak tersebut ke *priority queue* dengan mencatat *cost* dan urutan dibangkitkannya.

B. Checklist Keberjalanan Program

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

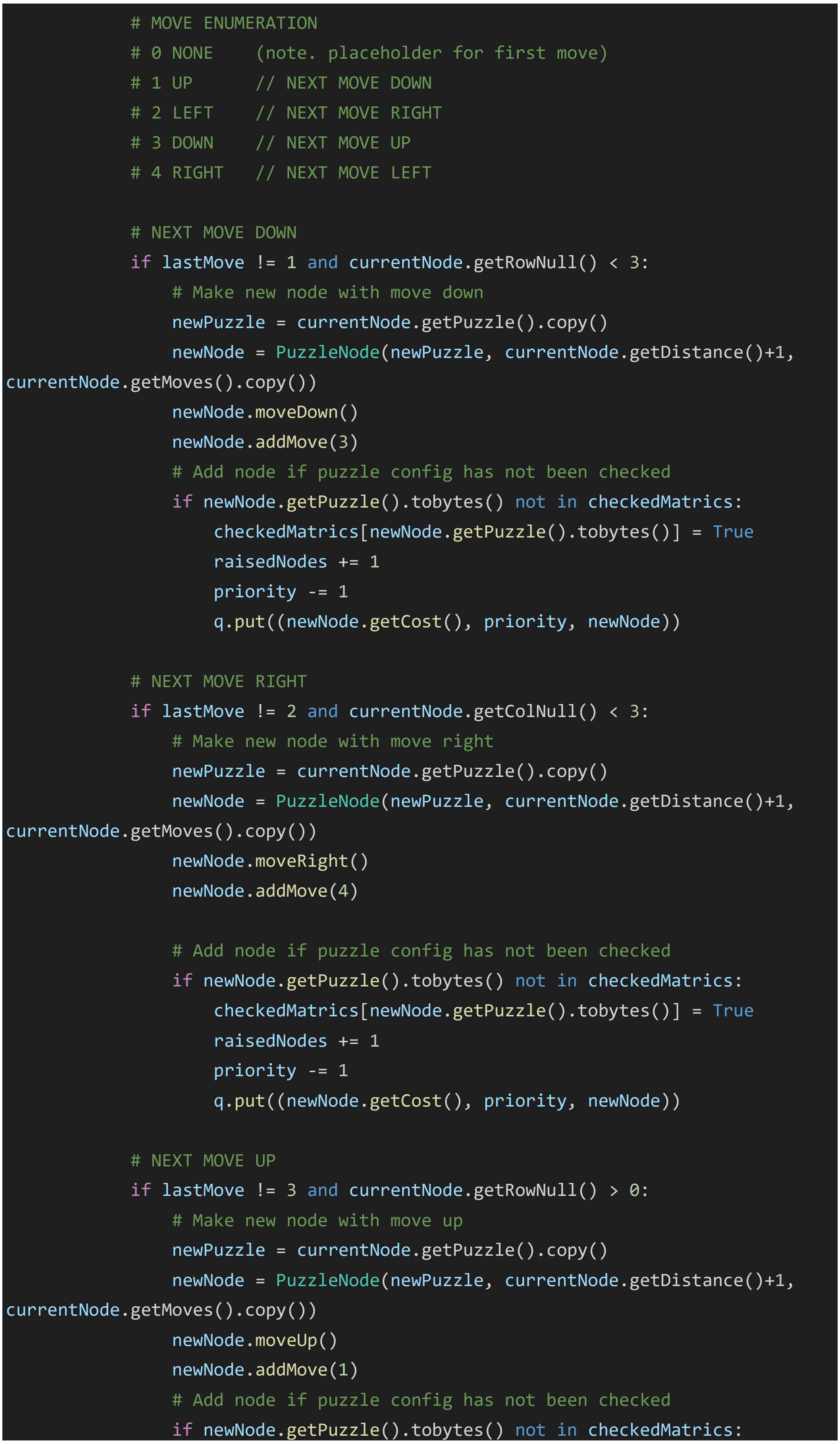
C. Kode Program dalam Python

Program ditulis pada bahasa Python secara modular, terdiri atas lima modul serta satu file *main.py* pada folder *src.*

1. Algorithm.py

Berisi fungsi algoritma *branch and bound* 15-Puzzle yang membuat sebuah *priority queue* dari konfigurasi puzzle awal yang sudah dibangkitkan secara acak atau melalui file. Fungsi algoritma tersebut mengembalikan simpul solusi yang berisi langkah-langkah penyelesaian dan banyaknya simpul yang dibangkitkan pada algoritma.

```
import numpy as np
from queue import PriorityQueue
from PuzzleNode import *


def Algorithm(Puzzle):
    # Make dict to check if puzzle config has been checked or not
    checkedMatrics = {}
    # Add root puzzle config to checkedMatrics
    checkedMatrics[Puzzle.tobytes()] = True

    # counter to prioritize innermost node
    priority = 0

    # Make prioqueue of nodes to be checked
    # to prioritize cost then innermost node
    q = PriorityQueue()

    # Add root node to queue
    rootNode = PuzzleNode(Puzzle, 0, [0])
    q.put((rootNode.getCost(), priority, rootNode))

    # Counter for raised nodes
    raisedNodes = 1

    # Initialize matrix to check for solution
    solutionMatrix = np.matrix([[1,2,3,4], [5,6,7,8], [9,10,11,12],
[13,14,15,0]])

    foundSolution = False
    while not q.empty() and not foundSolution:
        # Get innermost node with lowest cost
        currentNode = q.get()[2]
        lastMove = currentNode.getLastMove()

        # Check if node puzzle is solution
        if (currentNode.comparePuzzle(solutionMatrix)):
            foundSolution = True
        else:
```

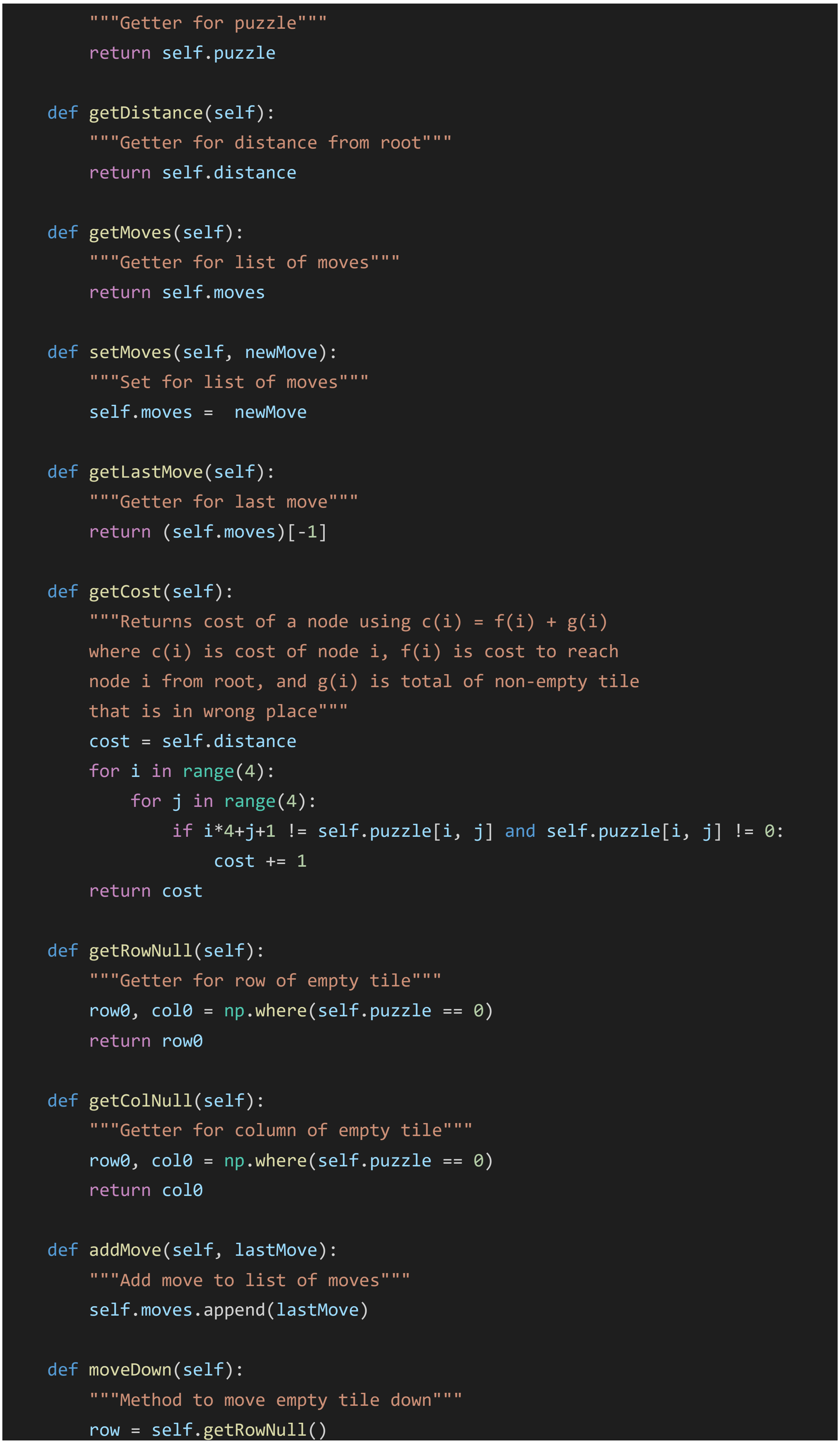


```
            # MOVE ENUMERATION
            # 0 NONE    (note. placeholder for first move)
            # 1 UP      // NEXT MOVE DOWN
            # 2 LEFT    // NEXT MOVE RIGHT
            # 3 DOWN    // NEXT MOVE UP
            # 4 RIGHT   // NEXT MOVE LEFT

            # NEXT MOVE DOWN
            if lastMove != 1 and currentNode.getRowNull() < 3:
                # Make new node with move down
                newPuzzle = currentNode.getPuzzle().copy()
                newNode = PuzzleNode(newPuzzle, currentNode.getDistance()+1, 
currentNode.getMoves().copy())
                newNode.moveDown()
                newNode.addMove(3)
                # Add node if puzzle config has not been checked
                if newNode.getPuzzle().tobytes() not in checkedMatrics:
                    checkedMatrics[newNode.getPuzzle().tobytes()] = True
                    raisedNodes += 1
                    priority -= 1
                    q.put((newNode.getCost(), priority, newNode))

            # NEXT MOVE RIGHT
            if lastMove != 2 and currentNode.getColNull() < 3:
                # Make new node with move right
                newPuzzle = currentNode.getPuzzle().copy()
                newNode = PuzzleNode(newPuzzle, currentNode.getDistance()+1, 
currentNode.getMoves().copy())
                newNode.moveRight()
                newNode.addMove(4)

                # Add node if puzzle config has not been checked
                if newNode.getPuzzle().tobytes() not in checkedMatrics:
                    checkedMatrics[newNode.getPuzzle().tobytes()] = True
                    raisedNodes += 1
                    priority -= 1
                    q.put((newNode.getCost(), priority, newNode))

            # NEXT MOVE UP
            if lastMove != 3 and currentNode.getRowNull() > 0:
                # Make new node with move up
                newPuzzle = currentNode.getPuzzle().copy()
                newNode = PuzzleNode(newPuzzle, currentNode.getDistance()+1, 
currentNode.getMoves().copy())
                newNode.moveUp()
                newNode.addMove(1)
                # Add node if puzzle config has not been checked
                if newNode.getPuzzle().tobytes() not in checkedMatrics:
                    checkedMatrics[newNode.getPuzzle().tobytes()] = True
                    raisedNodes += 1
                    priority -= 1
                    q.put((newNode.getCost(), priority, newNode))

            # NEXT MOVE LEFT
            if lastMove != 4 and currentNode.getColNull() > 0:
                # Make new node with move left
                newPuzzle = currentNode.getPuzzle().copy()
                newNode = PuzzleNode(newPuzzle, currentNode.getDistance()+1,
currentNode.getMoves().copy())
                newNode.moveLeft()
                newNode.addMove(2)
                # Add node if puzzle config has not been checked
                if newNode.getPuzzle().tobytes() not in checkedMatrics:
                    checkedMatrics[newNode.getPuzzle().tobytes()] = True
                    raisedNodes += 1
                    priority -= 1
                    q.put((newNode.getCost(), priority, newNode))

    currentNode.setMoves(currentNode.getMoves()[1::])
    return currentNode, raisedNodes
```

2. **PuzzleNode.py**

Berisi kelas PuzzleNode untuk merepresentasikan setiap simpul pada pencarian solusi algoritma. Setiap simpul memegang atribut konfigurasi puzzle, jarak dari root, dan langkah-langkah sebelumnya. Kelas mempunyai metode getter/setter yang diperlukan, penghitung cost dari ubin yang tidak di posisi benar, penggerak atas/bawah/kiri/kanan ubin kosong, pembanding konfigurasi puzzle, dan metode pendukung lainnya.

```
import numpy as np
from PuzzleGeneration import *

class PuzzleNode:
    """Class to represent a node in tree.
    A node contains the puzzle configuraton,
    distance from root, and list of moves"""
    def __init__(self, puzzle, distance, moves):
        """User-defined constructor for PuzzleNode class"""
        self.puzzle = puzzle
        self.distance = distance
        self.moves = moves

    def getPuzzle(self):
```



```python
        """Getter for puzzle"""
        return self.puzzle

    def getDistance(self):
        """Getter for distance from root"""
        return self.distance

    def getMoves(self):
        """Getter for list of moves"""
        return self.moves

    def setMoves(self, newMove):
        """Set for list of moves"""
        self.moves =  newMove

    def getLastMove(self):
        """Getter for last move"""
        return (self.moves)[-1]

    def getCost(self):
        """Returns cost of a node using c(i) = f(i) + g(i)
        where c(i) is cost of node i, f(i) is cost to reach
        node i from root, and g(i) is total of non-empty tile
        that is in wrong place"""
        cost = self.distance
        for i in range(4):
            for j in range(4):
                if i*4+j+1 != self.puzzle[i, j] and self.puzzle[i, j] != 0:
                    cost += 1
        return cost

    def getRowNull(self):
        """Getter for row of empty tile"""
        row0, col0 = np.where(self.puzzle == 0)
        return row0

    def getColNull(self):
        """Getter for column of empty tile"""
        row0, col0 = np.where(self.puzzle == 0)
        return col0

    def addMove(self, lastMove):
        """Add move to list of moves"""
        self.moves.append(lastMove)

    def moveDown(self):
        """Method to move empty tile down"""
        row = self.getRowNull()
        col = self.getColNull()
        self.puzzle[row, col] = self.puzzle[row+1, col]
        self.puzzle[row+1, col] = 0

    def moveRight(self):
        """Method to move empty tile right"""
        row = self.getRowNull()
        col = self.getColNull()
        self.puzzle[row, col] = self.puzzle[row, col+1]
        self.puzzle[row, col+1] = 0

    def moveUp(self):
        """Method to move empty tile up"""
        row = self.getRowNull()
        col = self.getColNull()
        self.puzzle[row, col] = self.puzzle[row-1, col]
        self.puzzle[row-1, col] = 0

    def moveLeft(self):
        """Method to move empty tile left"""
        row = self.getRowNull()
        col = self.getColNull()
        self.puzzle[row, col] = self.puzzle[row, col-1]
        self.puzzle[row, col-1] = 0

    def comparePuzzle(self, matrix):
        """Method to check if puzzle config is the same as matrix
        using binary comparison"""
        return self.puzzle.tobytes() == matrix.tobytes()

    def printNode(self):
        print("Puzzle:")
        PrintPuzzle(self.puzzle)
        print(f"Cost: {self.getCost()}")
        print(f"Distance: {self.distance}")
        print(f"Row: {self.getRowNull()} Col: {self.getColNull()}")
        print(f"Moves: {self.moves}")
        print()
```

3. Prerequisites.py

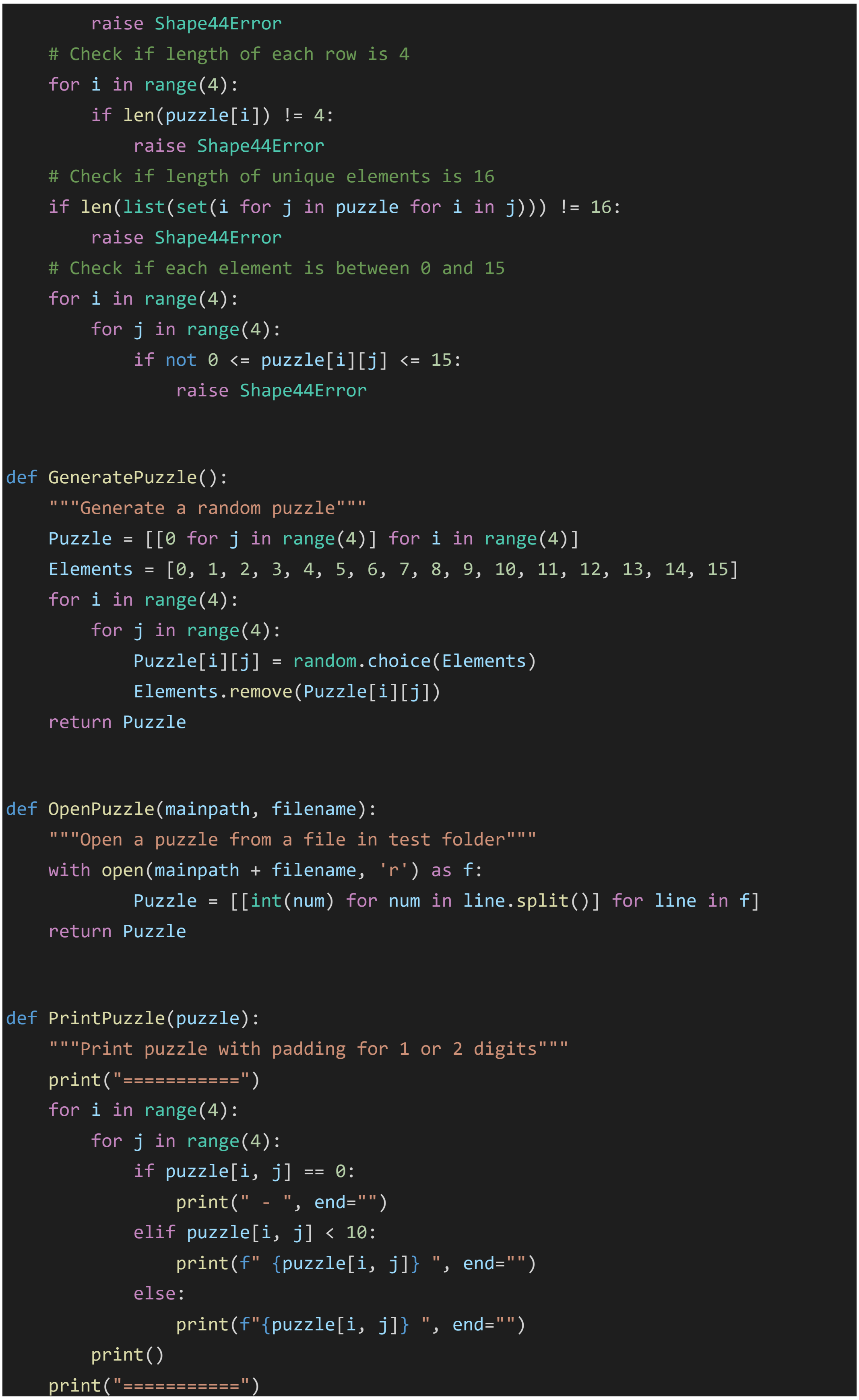
Berisi fungsi-fungsi yang menghitung Kurang(*i*) untuk setiap ubin *i* pada posisi awal, mencetak Kurang(*i*) untuk setiap ubin *i* tersebut, dan menghitung $\sum Kurang(i) + X$.

```
import numpy as np

def ArrayKurangI(puzzle):
    """Returns an array of index 0-15 to calculate Kurang(i)
    for each i tile"""
    # Initialize flattened array to iterate easier
    FlattenedPuzzle = (np.asarray(puzzle)).flatten()
    lengthFlattened = len(FlattenedPuzzle)
    KurangI = [0 for i in range(lengthFlattened)]

    # calculte sum of Kurang(i) for each i
    for i in range(lengthFlattened):
        # if i = 0, assume all next elements add up to Kurang(0)
        if FlattenedPuzzle[i] == 0:
            KurangI[FlattenedPuzzle[i]] += lengthFlattened - (i + 1)
        for j in range(i+1, lengthFlattened):
            if FlattenedPuzzle[j] < FlattenedPuzzle[i] and FlattenedPuzzle[j]
!= 0:
                KurangI[FlattenedPuzzle[i]] += 1
    return KurangI


def PrintTileKurangI(arr):
    """Print Kurang(i) for each i tile"""
    for i in range(1, len(arr)):
        print(f"Tile {i}: {arr[i]}")
    print(f"Empty tile: {arr[0]}")


def KurangIX(puzzle, arrKurangI):
    """calculate X based on 0 position"""
    if (np.where(puzzle == 0)[0] + np.where(puzzle == 0)[1]) % 2 == 1:
        return 1
    else:
        return 0
```

4. PuzzleGeneration.py

Berisi fungsi untuk membuat posisi awal puzzle dari acak atau dari file dan juga prosedur untuk mengecek apakah puzzle valid formatnya dan mencetak puzzle. Pengecek kevalidan puzzle akan *raise* error bila ada ketidakvalidan yang akan di-*catch* oleh program utama.

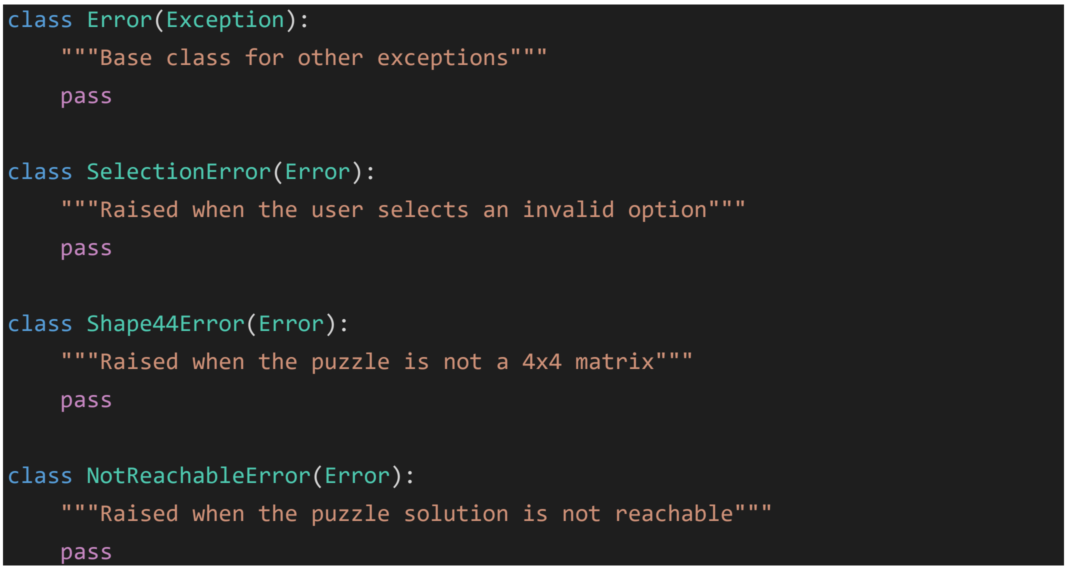
```
import random
from Exception import *

def CheckValidPuzzle(puzzle):
    """Check if a puzzle is considered valid"""

    # Check if rows of puzzle are 4
    if len(puzzle) != 4:
```



```
        raise Shape44Error
    # Check if length of each row is 4
    for i in range(4):
        if len(puzzle[i]) != 4:
            raise Shape44Error
    # Check if length of unique elements is 16
    if len(list(set(i for j in puzzle for i in j))) != 16:
        raise Shape44Error
    # Check if each element is between 0 and 15
    for i in range(4):
        for j in range(4):
            if not 0 <= puzzle[i][j] <= 15:
                raise Shape44Error


def GeneratePuzzle():
    """Generate a random puzzle"""
    Puzzle = [[0 for j in range(4)] for i in range(4)]
    Elements = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
    for i in range(4):
        for j in range(4):
            Puzzle[i][j] = random.choice(Elements)
            Elements.remove(Puzzle[i][j])
    return Puzzle


def OpenPuzzle(mainpath, filename):
    """Open a puzzle from a file in test folder"""
    with open(mainpath + filename, 'r') as f:
            Puzzle = [[int(num) for num in line.split()] for line in f]
    return Puzzle


def PrintPuzzle(puzzle):
    """Print puzzle with padding for 1 or 2 digits"""
    print("===========")
    for i in range(4):
        for j in range(4):
            if puzzle[i, j] == 0:
                print(" - ", end="")
            elif puzzle[i, j] < 10:
                print(f" {puzzle[i, j]} ", end="")
            else:
                print(f"{puzzle[i, j]} ", end="")
        print()
    print("===========")
```

5. Exception.py

Berisi kelas-kelas turunan *exception* untuk mencetak pesan bila terjadi error. Kelas-kelas tersebut digunakan untuk mengatasi error-error yang terdapat terutama pada pembangkitan puzzle dan juga pada input pengguna.

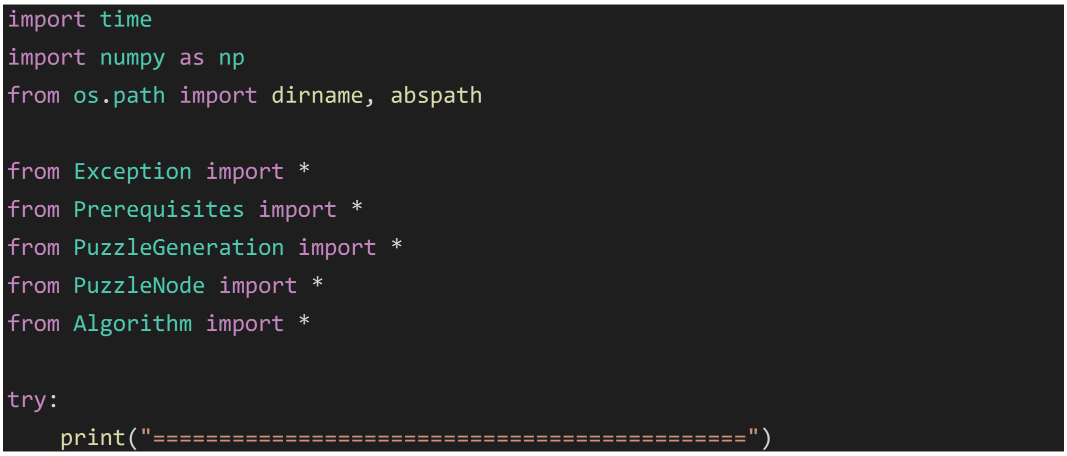


```python
class Error(Exception):
    """Base class for other exceptions"""
    pass

class SelectionError(Error):
    """Raised when the user selects an invalid option"""
    pass

class Shape44Error(Error):
    """Raised when the puzzle is not a 4x4 matrix"""
    pass

class NotReachableError(Error):
    """Raised when the puzzle solution is not reachable"""
    pass
```

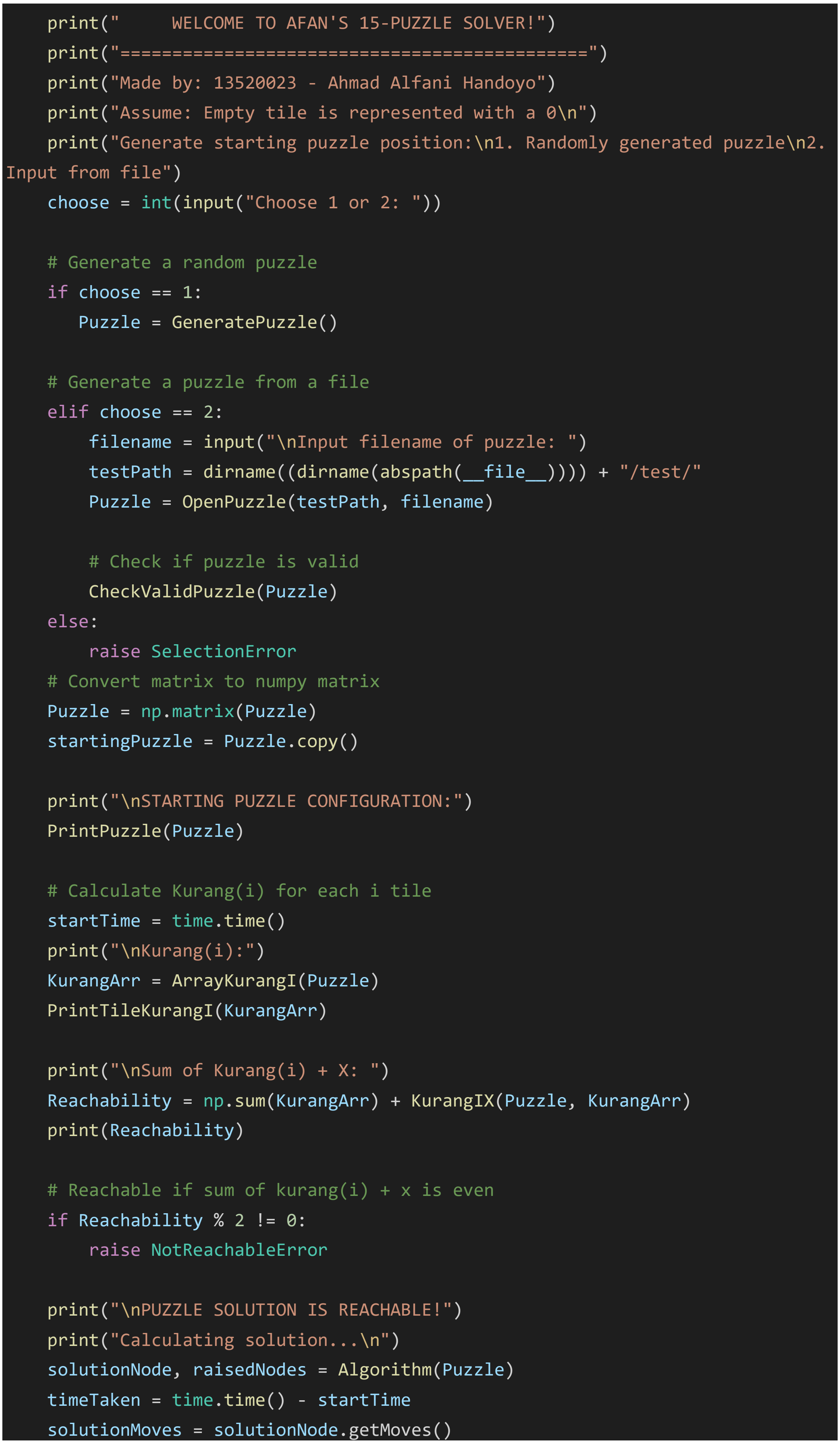
6. Main.py

File utama pada program memiliki urutan prosedur:

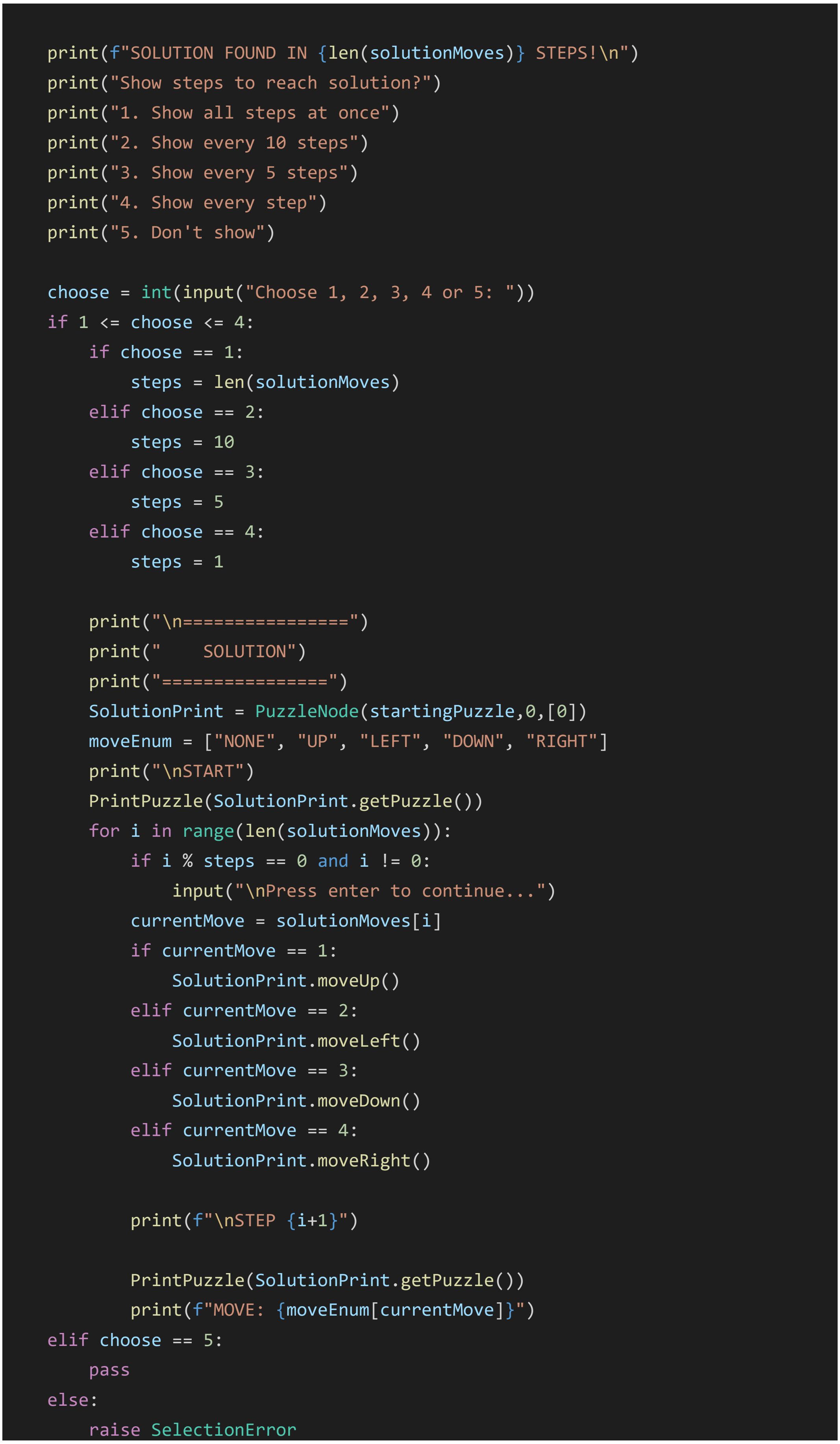
1. Membangkitkan puzzle sesuai pilihan pengguna secara acak atau input dari file
2. Menampilkan posisi awal puzzle
3. Menghitung nilai dari Kurang(i) setiap ubin i dan $\sum Kurang(i) + X$, mencetaknya, dan menentukan apakah puzzle dapat diselesaikan.
4. Menerapkan algoritma *branch and bound* untuk posisi awal puzzle
5. Menghitung waktu yang dibutuhkan untuk menjalankan algoritma
6. Mencetak langkah-langkah solusi, waktu yang dibutuhkan, dan jumlah simpul yang dibangkitkan.



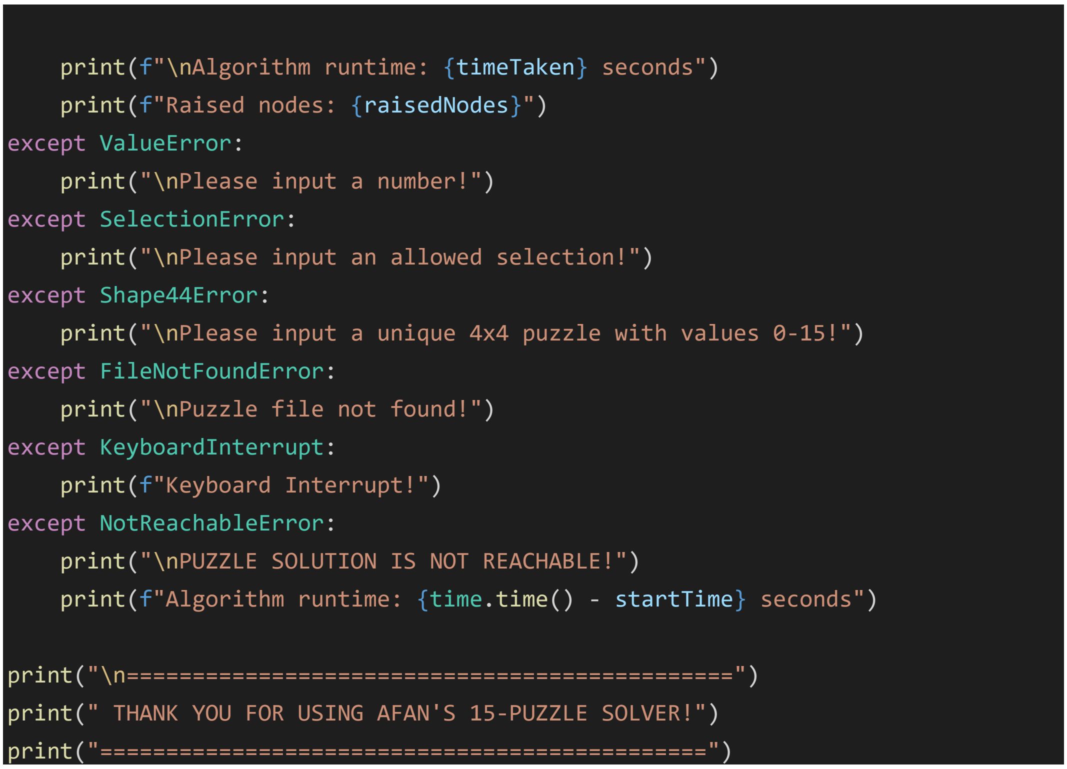
```python
import time
import numpy as np
from os.path import dirname, abspath

from Exception import *
from Prerequisites import *
from PuzzleGeneration import *
from PuzzleNode import *
from Algorithm import *

try:
    print("=============================================")
```



```
    print("     WELCOME TO AFAN'S 15-PUZZLE SOLVER!")
    print("=============================================")
    print("Made by: 13520023 - Ahmad Alfani Handoyo")
    print("Assume: Empty tile is represented with a 0\n")
    print("Generate starting puzzle position:\n1. Randomly generated puzzle\n2. 
Input from file")
    choose = int(input("Choose 1 or 2: "))

    # Generate a random puzzle
    if choose == 1:
       Puzzle = GeneratePuzzle()

    # Generate a puzzle from a file
    elif choose == 2:
        filename = input("\nInput filename of puzzle: ")
        testPath = dirname((dirname(abspath(__file__)))) + "/test/"
        Puzzle = OpenPuzzle(testPath, filename)

        # Check if puzzle is valid
        CheckValidPuzzle(Puzzle)
    else:
        raise SelectionError
    # Convert matrix to numpy matrix
    Puzzle = np.matrix(Puzzle)
    startingPuzzle = Puzzle.copy()

    print("\nSTARTING PUZZLE CONFIGURATION:")
    PrintPuzzle(Puzzle)

    # Calculate Kurang(i) for each i tile
    startTime = time.time()
    print("\nKurang(i):")
    KurangArr = ArrayKurangI(Puzzle)
    PrintTileKurangI(KurangArr)

    print("\nSum of Kurang(i) + X: ")
    Reachability = np.sum(KurangArr) + KurangIX(Puzzle, KurangArr)
    print(Reachability)

    # Reachable if sum of kurang(i) + x is even
    if Reachability % 2 != 0:
        raise NotReachableError

    print("\nPUZZLE SOLUTION IS REACHABLE!")
    print("Calculating solution...\n")
    solutionNode, raisedNodes = Algorithm(Puzzle)
    timeTaken = time.time() - startTime
    solutionMoves = solutionNode.getMoves()
```



```
print(f"SOLUTION FOUND IN {len(solutionMoves)} STEPS!\n")
print("Show steps to reach solution?")
print("1. Show all steps at once")
print("2. Show every 10 steps")
print("3. Show every 5 steps")
print("4. Show every step")
print("5. Don't show")

choose = int(input("Choose 1, 2, 3, 4 or 5: "))
if 1 <= choose <= 4:
    if choose == 1:
        steps = len(solutionMoves)
    elif choose == 2:
        steps = 10
    elif choose == 3:
        steps = 5
    elif choose == 4:
        steps = 1

    print("\n================")
    print("    SOLUTION")
    print("================")
    SolutionPrint = PuzzleNode(startingPuzzle,0,[0])
    moveEnum = ["NONE", "UP", "LEFT", "DOWN", "RIGHT"]
    print("\nSTART")
    PrintPuzzle(SolutionPrint.getPuzzle())
    for i in range(len(solutionMoves)):
        if i % steps == 0 and i != 0:
            input("\nPress enter to continue...")
        currentMove = solutionMoves[i]
        if currentMove == 1:
            SolutionPrint.moveUp()
        elif currentMove == 2:
            SolutionPrint.moveLeft()
        elif currentMove == 3:
            SolutionPrint.moveDown()
        elif currentMove == 4:
            SolutionPrint.moveRight()

        print(f"\nSTEP {i+1}")

        PrintPuzzle(SolutionPrint.getPuzzle())
        print(f"MOVE: {moveEnum[currentMove]}")
elif choose == 5:
    pass
else:
    raise SelectionError
```



```
    print(f"\nAlgorithm runtime: {timeTaken} seconds")
    print(f"Raised nodes: {raisedNodes}")
except ValueError:
    print("\nPlease input a number!")
except SelectionError:
    print("\nPlease input an allowed selection!")
except Shape44Error:
    print("\nPlease input a unique 4x4 puzzle with values 0-15!")
except FileNotFoundError:
    print("\nPuzzle file not found!")
except KeyboardInterrupt:
    print(f"Keyboard Interrupt!")
except NotReachableError:
    print("\nPUZZLE SOLUTION IS NOT REACHABLE!")
    print(f"Algorithm runtime: {time.time() - startTime} seconds")

print("\n==============================================")
print(" THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!")
print("==============================================")
```

D. Berkas Teks Contoh Instansiasi 5 Buah Persoalan 15-Puzzle

Pembangkitan konfigurasi awal puzzle dapat secara acak atau melalui input file. Bila melalui input file, maka konfigurasi awal puzzle dituliskan pada file berformat .txt dan dimasukkan ke folder *test*. Ubin kosong direpresentasikan dengan angka 0.

Dalam menguji coba program terdapat 5 *test case*. Berikut isi teks 5 *test case* tersebut sebagai contoh cara instansiasi input persoalan 15-Puzzle.

1. unsolvable1.txt

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

2. unsolvable2.txt

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

3. solvable1_15.txt

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

4. solvable2_20.txt

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

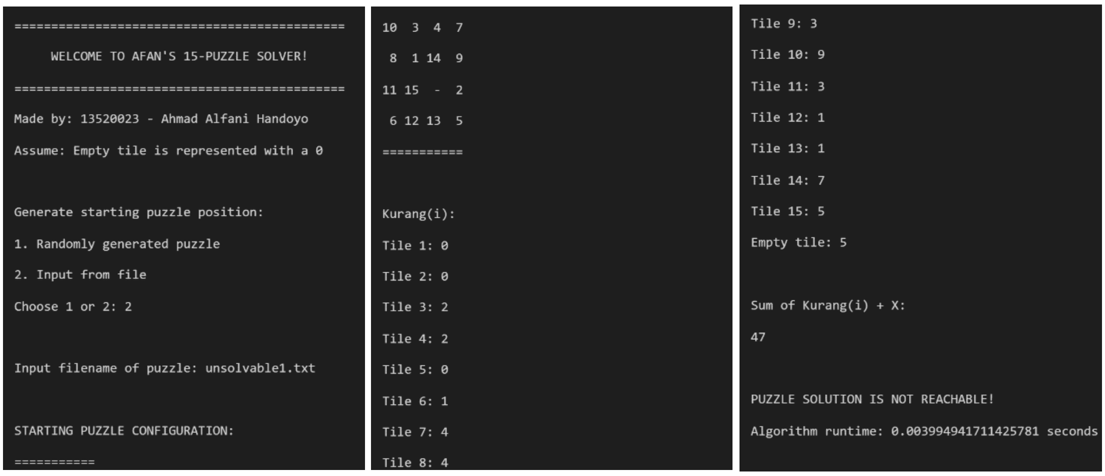
5. solvable3_25.txt

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

E. *Screenshot* Input dan Output Program

Program dijalankan dengan menjalankan *Main.py* pada folder *src*. Berikut merupakan input dan output pengujian 5 *test case* yang terdapat di atas.

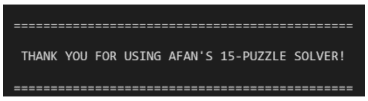
1. Kasus tidak dapat diselesaikan 1 *(unsolvable1.txt)*



```
============================================

     WELCOME TO AFAN'S 15-PUZZLE SOLVER!

============================================

Made by: 13520023 - Ahmad Alfani Handoyo

Assume: Empty tile is represented with a 0


Generate starting puzzle position:

1. Randomly generated puzzle

2. Input from file

Choose 1 or 2: 2


Input filename of puzzle: unsolvable1.txt


STARTING PUZZLE CONFIGURATION:

===========
```

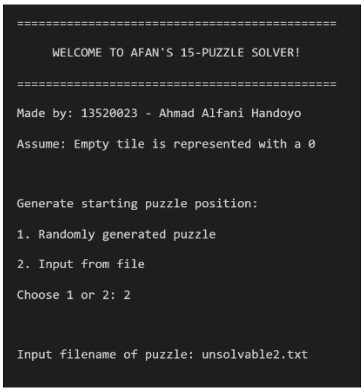
```
10  3  4  7

 8  1 14  9

11 15  -  2

 6 12 13  5

===========


Kurang(i):

Tile 1: 0

Tile 2: 0

Tile 3: 2

Tile 4: 2

Tile 5: 0

Tile 6: 1

Tile 7: 4

Tile 8: 4
```

```
Tile 9: 3

Tile 10: 9

Tile 11: 3

Tile 12: 1

Tile 13: 1

Tile 14: 7

Tile 15: 5

Empty tile: 5


Sum of Kurang(i) + X:

47


PUZZLE SOLUTION IS NOT REACHABLE!

Algorithm runtime: 0.003994941711425781 seconds
```

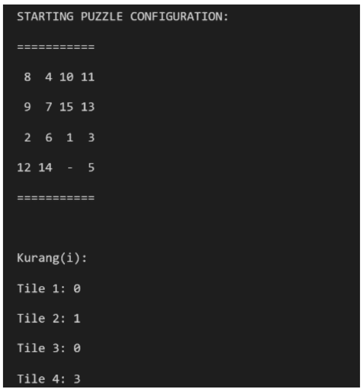


```
============================================

 THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!

============================================
```

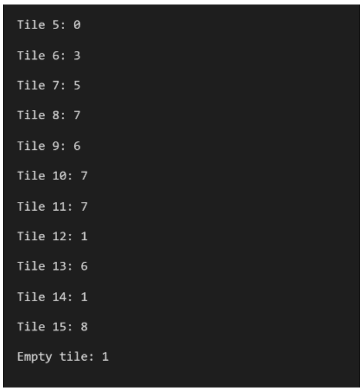
2. Kasus tidak dapat diselesaikan 2 *(unsolvable2.txt)*



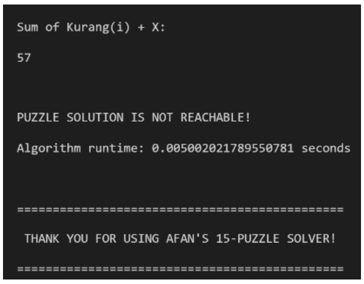
```
===========================================

     WELCOME TO AFAN'S 15-PUZZLE SOLVER!

===========================================

Made by: 13520023 - Ahmad Alfani Handoyo

Assume: Empty tile is represented with a 0


Generate starting puzzle position:

1. Randomly generated puzzle

2. Input from file

Choose 1 or 2: 2


Input filename of puzzle: unsolvable2.txt
```



```
STARTING PUZZLE CONFIGURATION:

==========

 8  4 10 11

 9  7 15 13

 2  6  1  3

12 14  -  5

==========


Kurang(i):

Tile 1: 0

Tile 2: 1

Tile 3: 0

Tile 4: 3
```

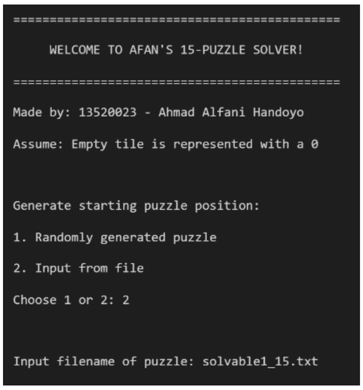


```
Tile 5: 0

Tile 6: 3

Tile 7: 5

Tile 8: 7

Tile 9: 6

Tile 10: 7

Tile 11: 7

Tile 12: 1

Tile 13: 6

Tile 14: 1

Tile 15: 8

Empty tile: 1
```

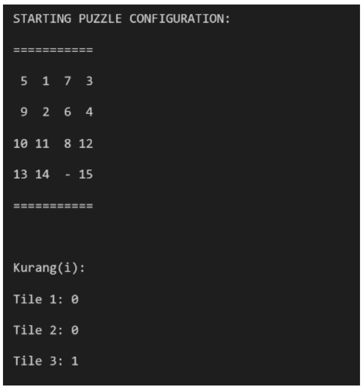


```
Sum of Kurang(i) + X:

57


PUZZLE SOLUTION IS NOT REACHABLE!

Algorithm runtime: 0.005002021789550781 seconds


============================================

 THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!

============================================
```

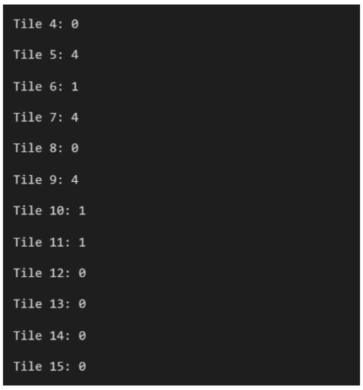
3. Kasus dapat diselesaikan, 15 langkah *(solvable1_15.txt)*



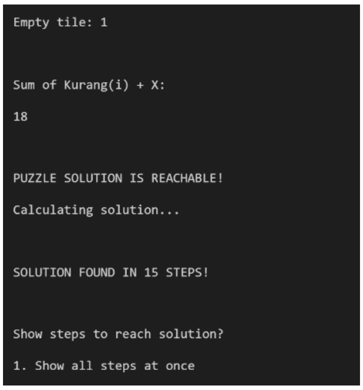
```
===========================================

     WELCOME TO AFAN'S 15-PUZZLE SOLVER!

===========================================

Made by: 13520023 - Ahmad Alfani Handoyo

Assume: Empty tile is represented with a 0


Generate starting puzzle position:

1. Randomly generated puzzle

2. Input from file

Choose 1 or 2: 2


Input filename of puzzle: solvable1_15.txt
```



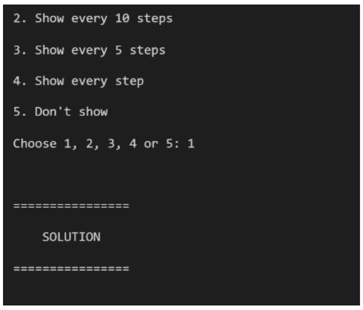
```
STARTING PUZZLE CONFIGURATION:

==========

 5  1  7  3

 9  2  6  4

10 11  8 12

13 14  - 15

==========


Kurang(i):

Tile 1: 0

Tile 2: 0

Tile 3: 1
```



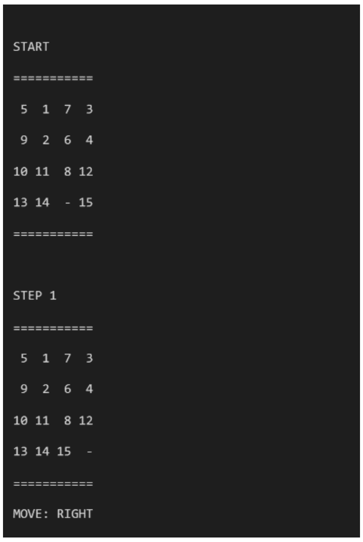
```
Tile 4: 0

Tile 5: 4

Tile 6: 1

Tile 7: 4

Tile 8: 0

Tile 9: 4

Tile 10: 1

Tile 11: 1

Tile 12: 0

Tile 13: 0

Tile 14: 0

Tile 15: 0
```



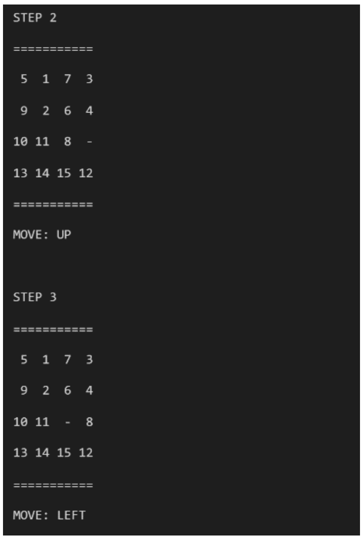
```
Empty tile: 1


Sum of Kurang(i) + X:

18


PUZZLE SOLUTION IS REACHABLE!

Calculating solution...


SOLUTION FOUND IN 15 STEPS!


Show steps to reach solution?

1. Show all steps at once
```



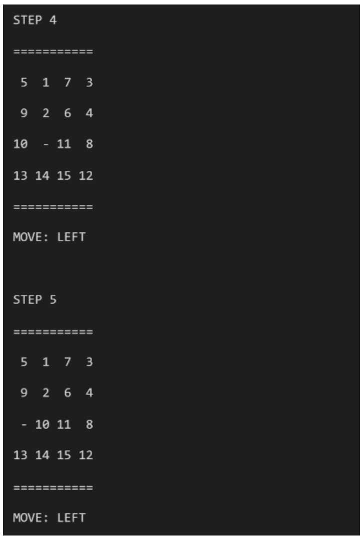
```
2. Show every 10 steps

3. Show every 5 steps

4. Show every step

5. Don't show

Choose 1, 2, 3, 4 or 5: 1


===============

    SOLUTION

===============
```



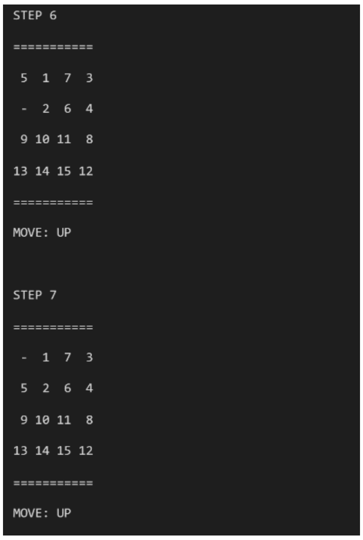
```
START
===========
 5  1  7  3
 9  2  6  4
10 11  8 12
13 14  - 15
===========


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



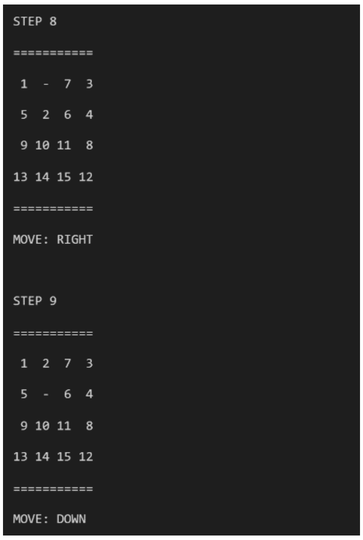
```
STEP 2
===========
 5  1  7  3
 9  2  6  4
10 11  8  -
13 14 15 12
===========
MOVE: UP


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



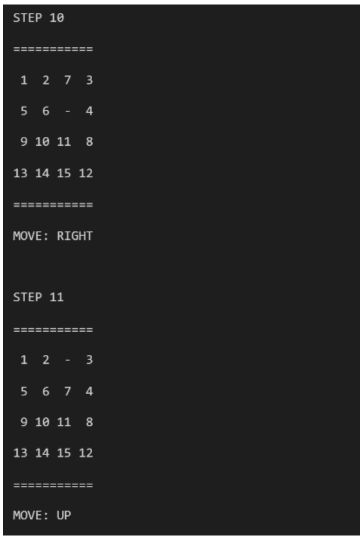
```
STEP 4
===========
 5  1  7  3
 9  2  6  4
10  - 11  8
13 14 15 12
===========
MOVE: LEFT


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



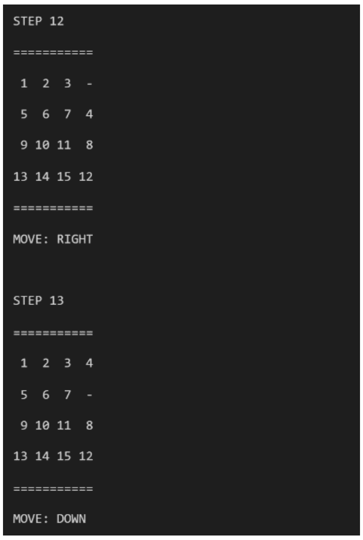
```
STEP 6
===========
 5  1  7  3
 -  2  6  4
 9 10 11  8
13 14 15 12
===========
MOVE: UP


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



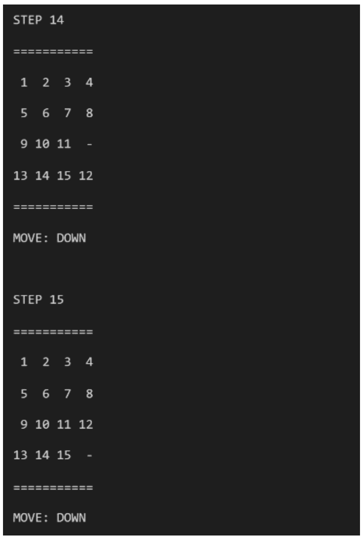
```
STEP 8
===========
 1  -  7  3
 5  2  6  4
 9 10 11  8
13 14 15 12
===========
MOVE: RIGHT


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



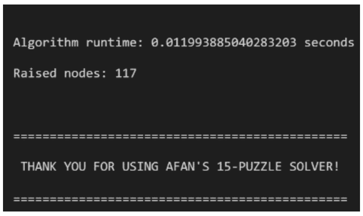
```
STEP 10
===========
 1  2  7  3
 5  6  -  4
 9 10 11  8
13 14 15 12
===========
MOVE: RIGHT


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



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


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

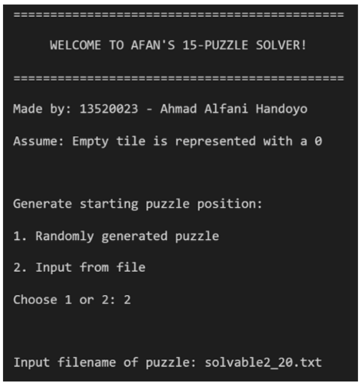


```
STEP 14
===========
 1  2  3  4
 5  6  7  8
 9 10 11  -
13 14 15 12
===========
MOVE: DOWN


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

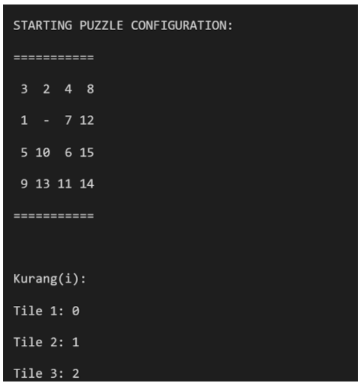


```
Algorithm runtime: 0.011993885040283203 seconds
Raised nodes: 117


============================================
 THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!
============================================
```

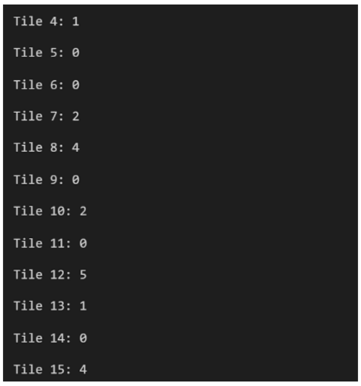
4. Kasus dapat diselesaikan, 20 langkah *(solvable2_20.txt)*



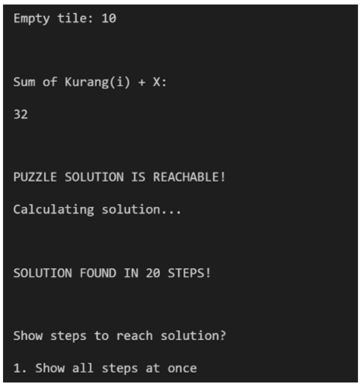
```
============================================

     WELCOME TO AFAN'S 15-PUZZLE SOLVER!

============================================

Made by: 13520023 - Ahmad Alfani Handoyo

Assume: Empty tile is represented with a 0


Generate starting puzzle position:

1. Randomly generated puzzle

2. Input from file

Choose 1 or 2: 2


Input filename of puzzle: solvable2_20.txt
```



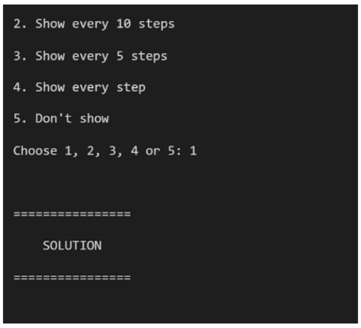
```
STARTING PUZZLE CONFIGURATION:

===========

 3  2  4  8

 1  -  7 12

 5 10  6 15

 9 13 11 14

===========


Kurang(i):

Tile 1: 0

Tile 2: 1

Tile 3: 2
```



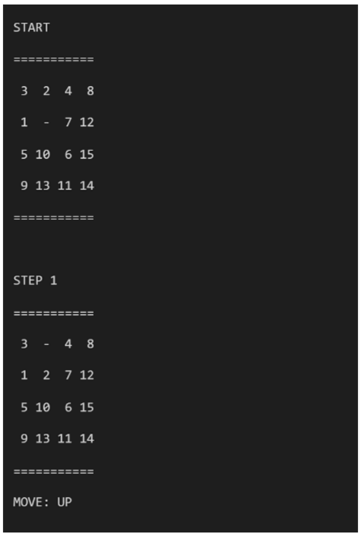
```
Tile 4: 1

Tile 5: 0

Tile 6: 0

Tile 7: 2

Tile 8: 4

Tile 9: 0

Tile 10: 2

Tile 11: 0

Tile 12: 5

Tile 13: 1

Tile 14: 0

Tile 15: 4
```



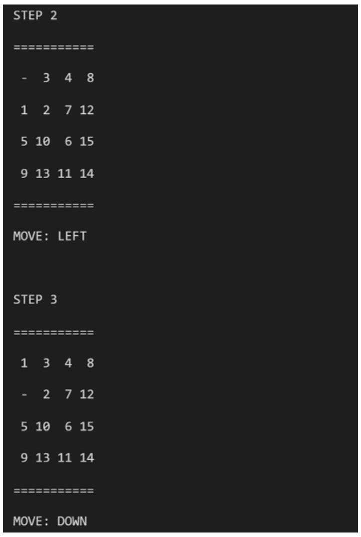
```
Empty tile: 10


Sum of Kurang(i) + X:

32


PUZZLE SOLUTION IS REACHABLE!

Calculating solution...


SOLUTION FOUND IN 20 STEPS!


Show steps to reach solution?

1. Show all steps at once
```



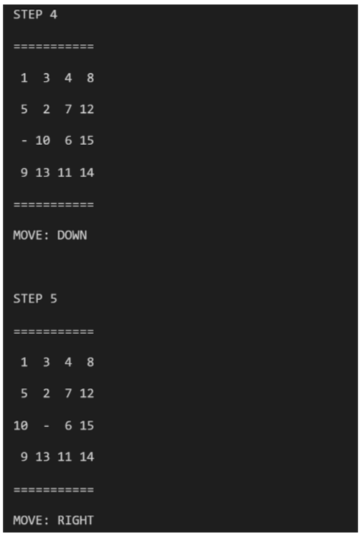
```
2. Show every 10 steps

3. Show every 5 steps

4. Show every step

5. Don't show

Choose 1, 2, 3, 4 or 5: 1


================

    SOLUTION

================
```



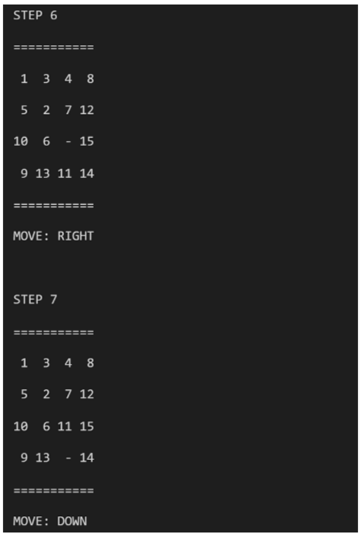
```
START

===========

 3  2  4  8

 1  -  7 12

 5 10  6 15

 9 13 11 14

===========


STEP 1

===========

 3  -  4  8

 1  2  7 12

 5 10  6 15

 9 13 11 14

===========

MOVE: UP
```



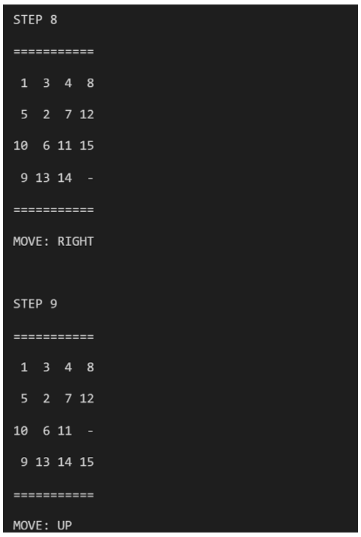
```
STEP 2

===========

 -  3  4  8

 1  2  7 12

 5 10  6 15

 9 13 11 14

===========

MOVE: LEFT


STEP 3

===========

 1  3  4  8

 -  2  7 12

 5 10  6 15

 9 13 11 14

===========

MOVE: DOWN
```



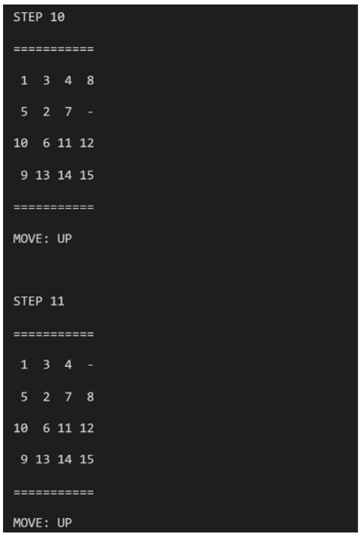
```
STEP 4

===========

 1  3  4  8

 5  2  7 12

 - 10  6 15

 9 13 11 14

===========

MOVE: DOWN


STEP 5

===========

 1  3  4  8

 5  2  7 12

10  -  6 15

 9 13 11 14

===========

MOVE: RIGHT
```



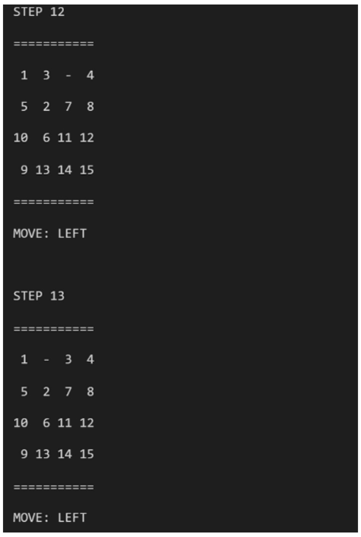
```
STEP 6
===========
 1  3  4  8
 5  2  7 12
10  6  - 15
 9 13 11 14
===========
MOVE: RIGHT


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



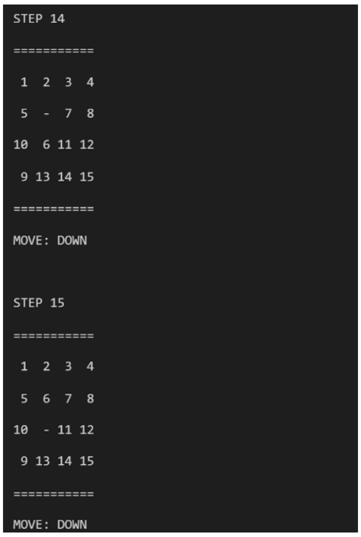
```
STEP 8
===========
 1  3  4  8
 5  2  7 12
10  6 11 15
 9 13 14  -
===========
MOVE: RIGHT


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



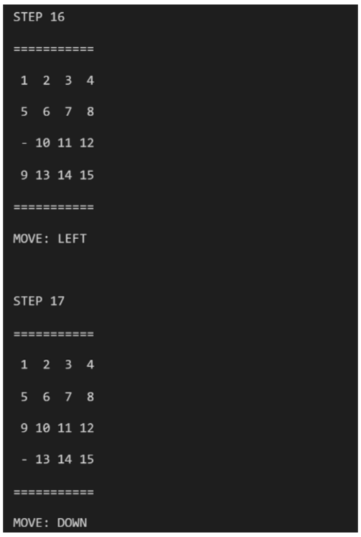
```
STEP 10
===========
 1  3  4  8
 5  2  7  -
10  6 11 12
 9 13 14 15
===========
MOVE: UP


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



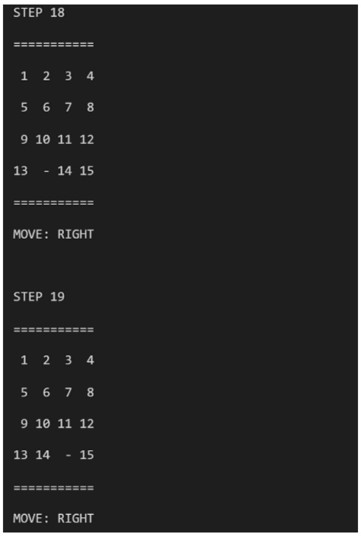
```
STEP 12
===========
 1  3  -  4
 5  2  7  8
10  6 11 12
 9 13 14 15
===========
MOVE: LEFT


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



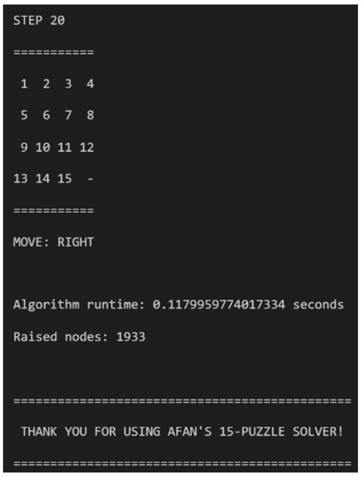
```
STEP 14
===========
 1  2  3  4
 5  -  7  8
10  6 11 12
 9 13 14 15
===========
MOVE: DOWN


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



```
STEP 16
===========
 1  2  3  4
 5  6  7  8
 - 10 11 12
 9 13 14 15
===========
MOVE: LEFT


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

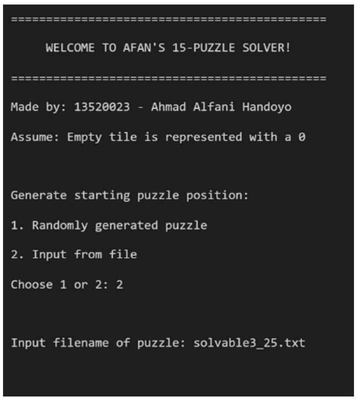


```
STEP 18
===========
 1  2  3  4
 5  6  7  8
 9 10 11 12
13  - 14 15
===========
MOVE: RIGHT


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

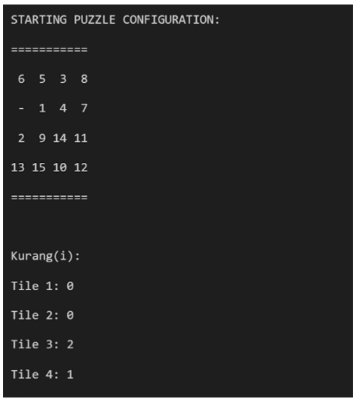


```
STEP 20
===========
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  -
===========
MOVE: RIGHT


Algorithm runtime: 0.1179959774017334 seconds
Raised nodes: 1933


============================================
 THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!
============================================
```

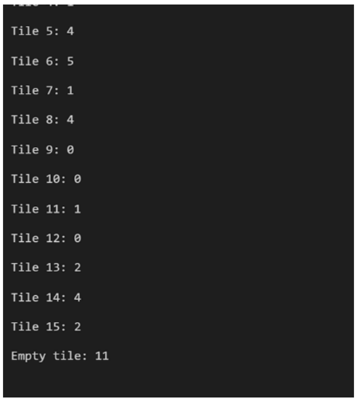
5. Kasus dapat diselesaikan, 25 langkah *(solvable3_25.txt)*



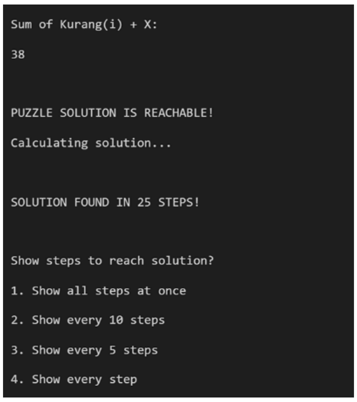
```
============================================

     WELCOME TO AFAN'S 15-PUZZLE SOLVER!

============================================

Made by: 13520023 - Ahmad Alfani Handoyo

Assume: Empty tile is represented with a 0


Generate starting puzzle position:

1. Randomly generated puzzle

2. Input from file

Choose 1 or 2: 2


Input filename of puzzle: solvable3_25.txt
```



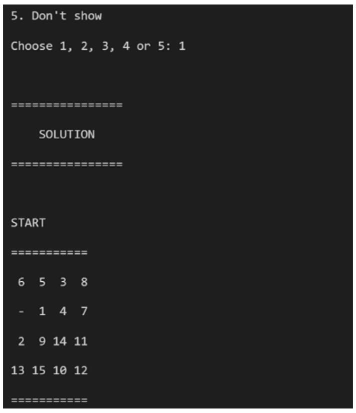
```
STARTING PUZZLE CONFIGURATION:

===========

 6  5  3  8

 -  1  4  7

 2  9 14 11

13 15 10 12

===========


Kurang(i):

Tile 1: 0

Tile 2: 0

Tile 3: 2

Tile 4: 1
```



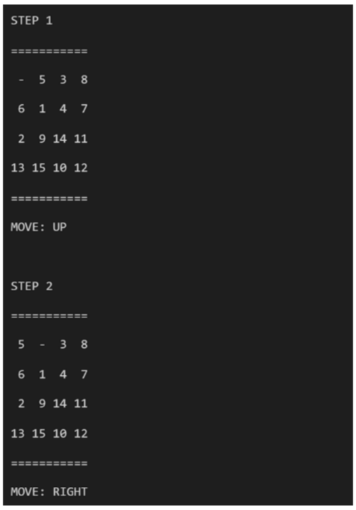
```
Tile 5: 4

Tile 6: 5

Tile 7: 1

Tile 8: 4

Tile 9: 0

Tile 10: 0

Tile 11: 1

Tile 12: 0

Tile 13: 2

Tile 14: 4

Tile 15: 2

Empty tile: 11
```



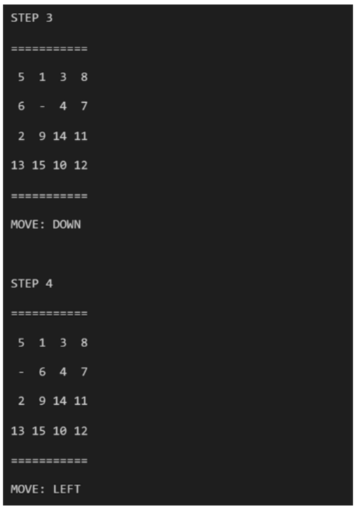
```
Sum of Kurang(i) + X:

38


PUZZLE SOLUTION IS REACHABLE!

Calculating solution...


SOLUTION FOUND IN 25 STEPS!


Show steps to reach solution?

1. Show all steps at once

2. Show every 10 steps

3. Show every 5 steps

4. Show every step
```



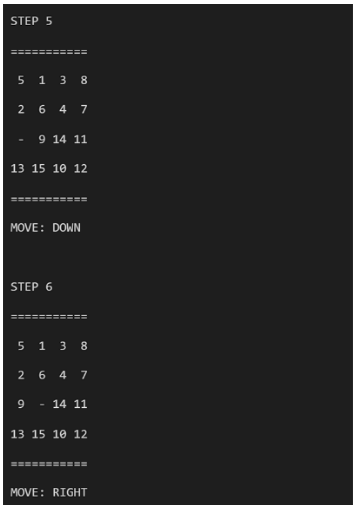
```
5. Don't show

Choose 1, 2, 3, 4 or 5: 1


================

    SOLUTION

================


START

===========

 6  5  3  8

 -  1  4  7

 2  9 14 11

13 15 10 12

===========
```



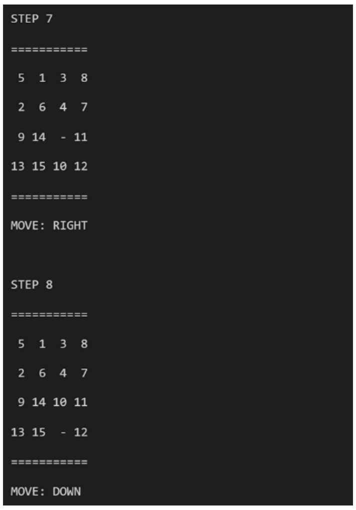
```
STEP 1

===========

 -  5  3  8

 6  1  4  7

 2  9 14 11

13 15 10 12

===========

MOVE: UP


STEP 2

===========

 5  -  3  8

 6  1  4  7

 2  9 14 11

13 15 10 12

===========

MOVE: RIGHT
```



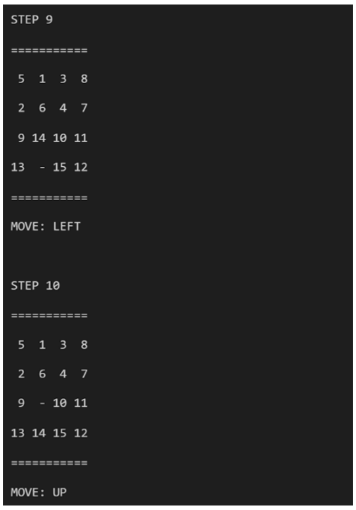
```
STEP 3

===========

 5  1  3  8

 6  -  4  7

 2  9 14 11

13 15 10 12

===========

MOVE: DOWN


STEP 4

===========

 5  1  3  8

 -  6  4  7

 2  9 14 11

13 15 10 12

===========

MOVE: LEFT
```



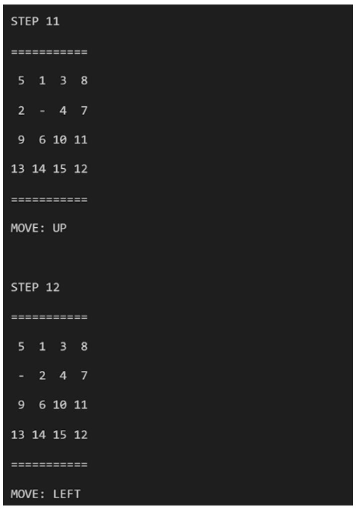
```
STEP 5

===========

 5  1  3  8

 2  6  4  7

 -  9 14 11

13 15 10 12

===========

MOVE: DOWN


STEP 6

===========

 5  1  3  8

 2  6  4  7

 9  - 14 11

13 15 10 12

===========

MOVE: RIGHT
```



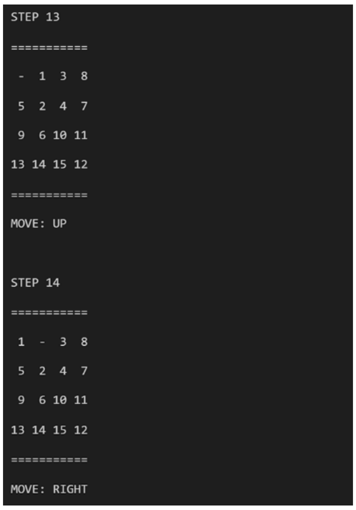
```
STEP 7
===========
 5  1  3  8
 2  6  4  7
 9 14  - 11
13 15 10 12
===========
MOVE: RIGHT

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



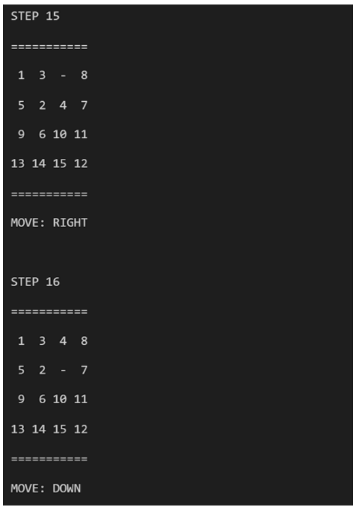
```
STEP 9
===========
 5  1  3  8
 2  6  4  7
 9 14 10 11
13  - 15 12
===========
MOVE: LEFT

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



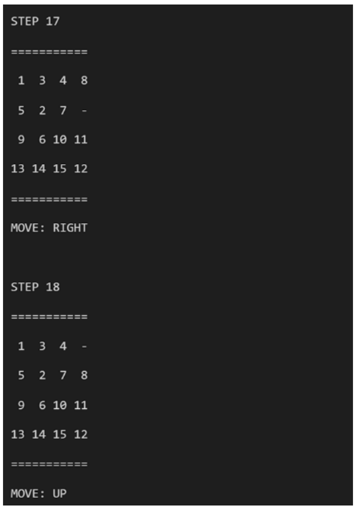
```
STEP 11
===========
 5  1  3  8
 2  -  4  7
 9  6 10 11
13 14 15 12
===========
MOVE: UP

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



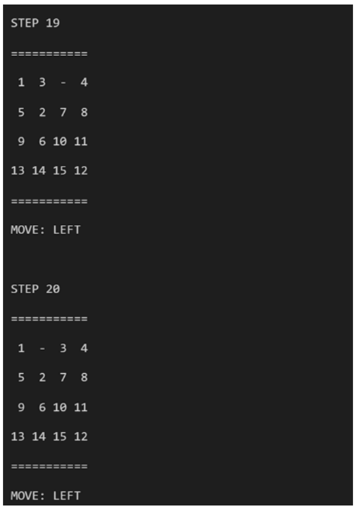
```
STEP 13
===========
 -  1  3  8
 5  2  4  7
 9  6 10 11
13 14 15 12
===========
MOVE: UP

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



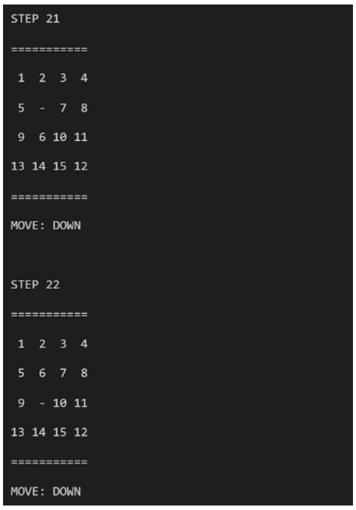
```
STEP 15
===========
 1  3  -  8
 5  2  4  7
 9  6 10 11
13 14 15 12
===========
MOVE: RIGHT

STEP 16
===========
 1  3  4  8
 5  2  -  7
 9  6 10 11
13 14 15 12
===========
MOVE: DOWN
```



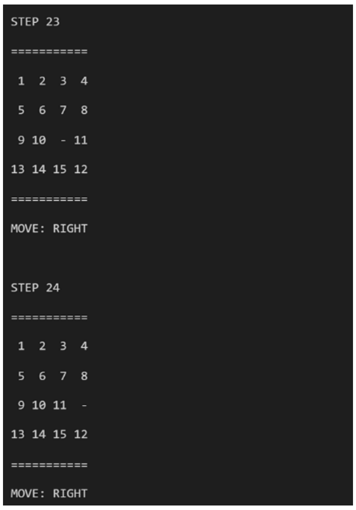
```
STEP 17
===========
 1  3  4  8
 5  2  7  -
 9  6 10 11
13 14 15 12
===========
MOVE: RIGHT

STEP 18
===========
 1  3  4  -
 5  2  7  8
 9  6 10 11
13 14 15 12
===========
MOVE: UP
```



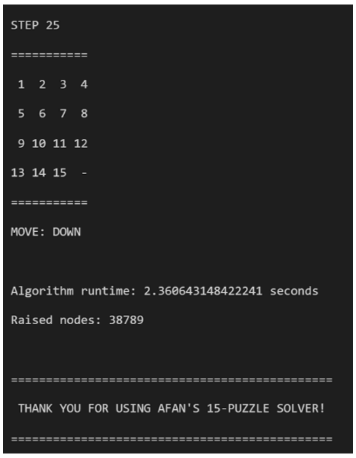
```
STEP 19
===========
 1  3  -  4
 5  2  7  8
 9  6 10 11
13 14 15 12
===========
MOVE: LEFT

STEP 20
===========
 1  -  3  4
 5  2  7  8
 9  6 10 11
13 14 15 12
===========
MOVE: LEFT
```



```
STEP 21
===========
 1  2  3  4
 5  -  7  8
 9  6 10 11
13 14 15 12
===========
MOVE: DOWN

STEP 22
===========
 1  2  3  4
 5  6  7  8
 9  - 10 11
13 14 15 12
===========
MOVE: DOWN
```



```
STEP 23
===========
 1  2  3  4
 5  6  7  8
 9 10  - 11
13 14 15 12
===========
MOVE: RIGHT

STEP 24
===========
 1  2  3  4
 5  6  7  8
 9 10 11  -
13 14 15 12
===========
MOVE: RIGHT
```



```
STEP 25

===========

 1  2  3  4

 5  6  7  8

 9 10 11 12

13 14 15  -

===========

MOVE: DOWN


Algorithm runtime: 2.360643148422241 seconds

Raised nodes: 38789


==============================================

 THANK YOU FOR USING AFAN'S 15-PUZZLE SOLVER!

==============================================
```

F. Alamat GitHub Program

Program dapat diunduh pada repository berikut:

https://github.com/blueguy42/Tucil3_13520023