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

Disusun Oleh:
Farnas Rozaan Iraqee (13520067)

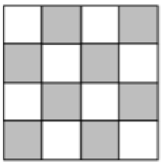
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021/2022

Algoritma *Branch and Bound*

Berikut ini adalah cara kerja program penyelesaian 15-*Puzzle* yang dibuat dengan algoritma *Branch and Bound*. Pertama, kita harus menentukan susunan awal dari *puzzle* seperti apa, dalam program ini terdapat dua pilihan dalam penentuan susunan awal dari *puzzle*, yaitu dengan menggunakan file konfigurasi yang berupa .txt atau dengan membangitkannya secara acak. Kemudian, kita akan menentukan apakah *puzzle* dapat diselesaikan apabila susunan awalnya seperti yang sudah ditentukan di awal. Kondisi di atas ditentukan dengan menggunakan formula sebagai berikut:

$$\sum_{i=1}^{16} KURANG(i) + X$$

Fungsi KURANG(i) adalah banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa. Kemudian, X=1 jika ubin kosong pada susunan awal ada pada sel yg diarsir.



Selanjutnya, kita akan mulai menyelesaikan 15-*puzzle*-nya dengan terlebih dahulu membuat *priority queue* yang berfungsi untuk menentukan simpul mana yang akan diproses terlebih dahulu. *Priority* yang digunakan dalam *priority queue* adalah ongkos taksiran minimal dari simpul. Misal, ongkos taksiran dari simpul P adalah sebagai berikut:

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

Dengan $f(P)$ adalah panjang lintasan dari simpul akar ke simpul P dan $\hat{g}(P)$ adalah jumlah ubin tidak kosong yang tidak terdapat dalam susunan akhir.

Kemudian, kita akan menentukan susunan *puzzle* seperti apa yang mungkin dicapai. Terdapat 4 susunan *puzzle* yang mungkin dicapai dalam kasus ini. Kemungkinan susunan tersebut adalah sebagai berikut:

1. Susunan *puzzle* dengan ubin kosong dipindah ke atas
2. Susunan *puzzle* dengan ubin kosong dipindah ke bawah
3. Susunan *puzzle* dengan ubin kosong dipindah ke kiri
4. Susunan *puzzle* dengan ubin kosong dipindah ke kanan

Keempat kemungkinan di atas tidak dapat dicapai apabila posisi ubin kosong sudah terletak di bagian paling atas, paling bawah, paling kiri , atau paling kanan. Apabila susunan *puzzle* dapat dicapai, maka pada pohon ruang status akan dibangkitkan simpul yang merepresentasikan susunan *puzzle* tersebut dan juga susunan *puzzle* tersebut di-*enqueue* ke *priority queue*, sebaliknya tidak. Namun, jika arah perpindahan merupakan kebalikan dari arah perpindahan susunan sebelumnya, simpul yang merepresentasikan susunan *puzzle* tersebut tidak akan dibangkitkan,. Selanjutnya, akan diproses susunan *puzzle* yang berada pada posisi terdepan pada *priority queue*. Langkah di atas akan diulangi hingga susunan *puzzle* mencapai susunan akhir.

Source Code Program dalam Bahasa Python

Program ini terdiri 4 file .py, yaitu puzzle.py, prioqueue.py, statespacetree.py, dan main.py. Berikut adalah *source code* isi dari keempat file tersebut:

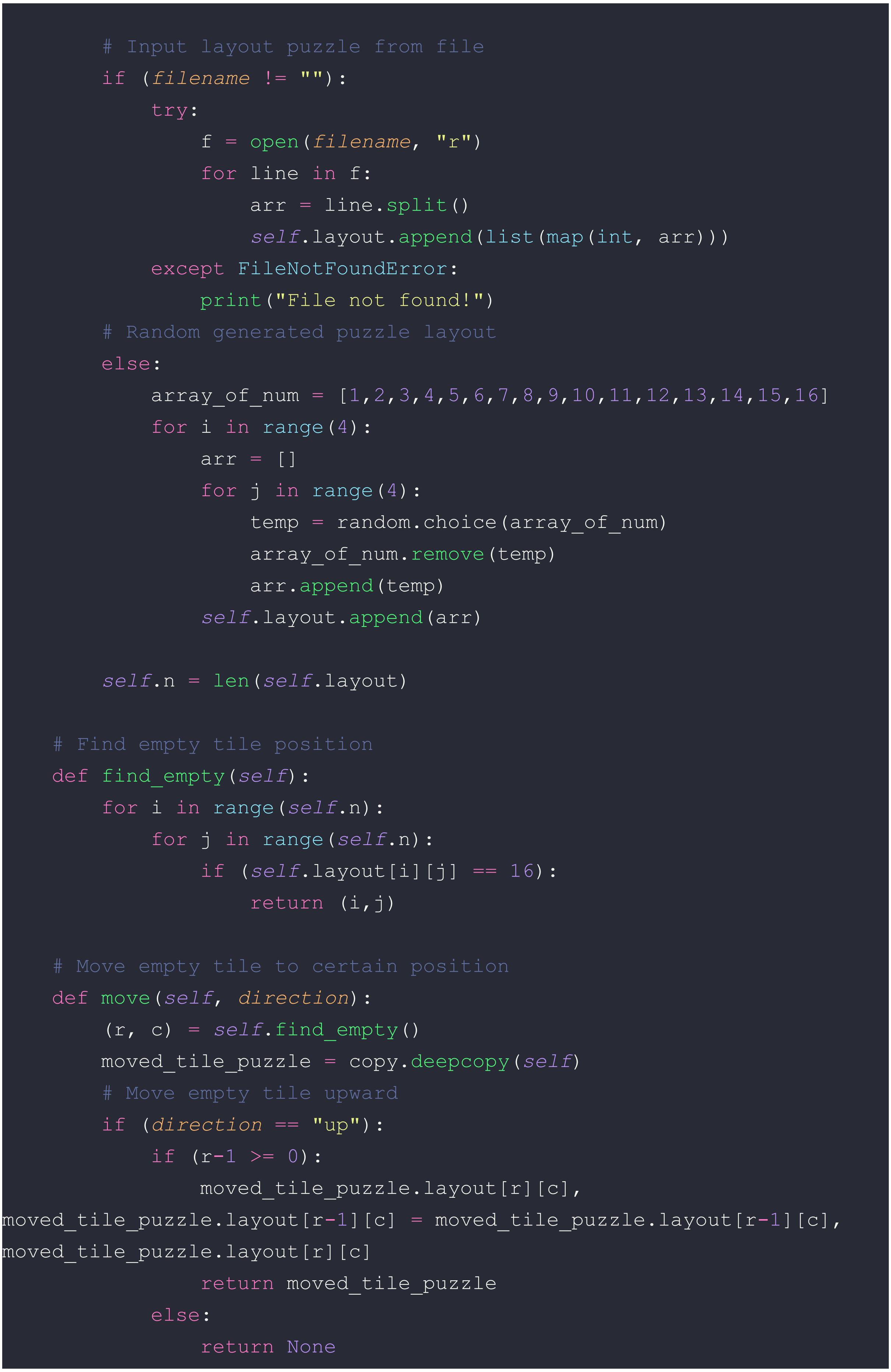
1. main.py

```python
import time
from puzzle import Puzzle
from statespacetree import StateSpaceTree
from prioqueue import PriorityQueue

# Check whether the user wants to run the program by using an external file or not
check = input("Do you wanna use external file to run this program? (y/n): ")

if (check == "y"):
    filename = input("Enter filename (.txt): ")
    tree = StateSpaceTree(Puzzle("../test/" + filename))
elif (check == "n"):
    tree = StateSpaceTree(Puzzle())
else:
    while (check != "y" and check != "n"):
        print("Invalid input! Try again!")
        check = input("Do you wanna use external file to run this program? (y/n): ")
    if (check == "y"):
        filename = input("Input filename (.txt): ")
        tree = StateSpaceTree(Puzzle("../test/" + filename))
    elif (check == "n"):
        tree = StateSpaceTree(Puzzle())

# Print the start state of puzzle
print()
print("This is the start state of puzzle")
tree.node.print_puzzle()
print()

# Check whether the puzzle is solvable or not
```

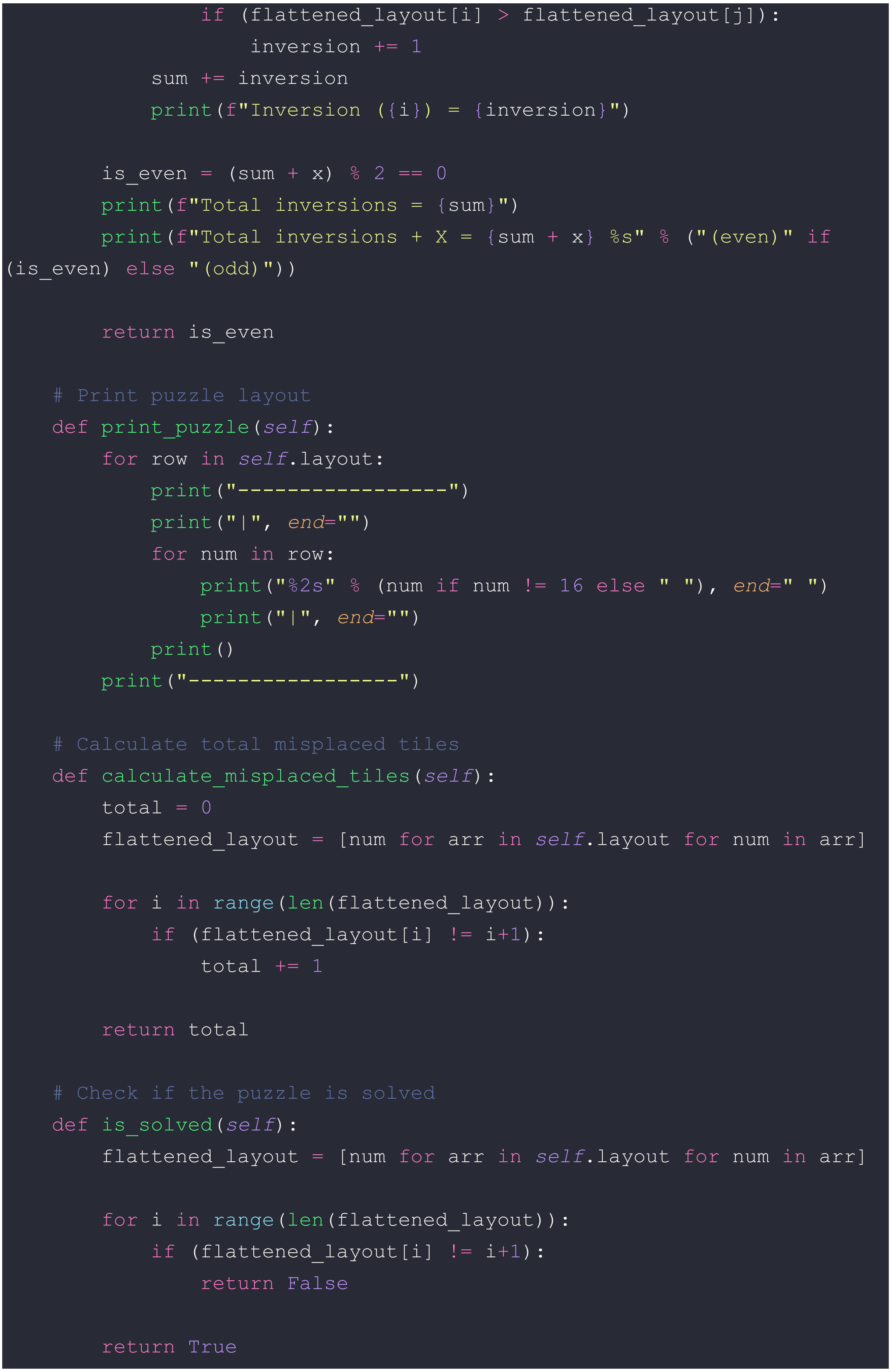
```python
if (not tree.node.is_solvable()):
    print()
    print("This puzzle isn't solvable. Exit program...")
    exit()

print()
print("This puzzle is solvable. Continuing program...")
print()

# Create priority queue for search needs
prioqueue = PriorityQueue()

# Enqueue start state of state space tree to priority queue
prioqueue.enqueue(tree)

# Start timer
start = time.process_time()

# Solving puzzle
final_state = prioqueue.solve()

# Stop timer
stop = time.process_time()

# Solution
final_state.show_solution()

# Total generated node
print("Total generated node =", StateSpaceTree.node_generated)

# Time elapsed
time_elapsed = stop - start
print(f"Time elapsed = {time_elapsed} s or {time_elapsed * 10**3} ms")
```

2. puzzle.py

```python
import copy
import random

class Puzzle:
    # Constructor
    def __init__(self, filename=""):
        self.layout = []
        self.n = 0
```



```
        # Input layout puzzle from file
        if (filename != ""):
            try:
                f = open(filename, "r")
                for line in f:
                    arr = line.split()
                    self.layout.append(list(map(int, arr)))
            except FileNotFoundError:
                print("File not found!")
        # Random generated puzzle layout
        else:
            array_of_num = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
            for i in range(4):
                arr = []
                for j in range(4):
                    temp = random.choice(array_of_num)
                    array_of_num.remove(temp)
                    arr.append(temp)
                self.layout.append(arr)

        self.n = len(self.layout)

    # Find empty tile position
    def find_empty(self):
        for i in range(self.n):
            for j in range(self.n):
                if (self.layout[i][j] == 16):
                    return (i,j)

    # Move empty tile to certain position
    def move(self, direction):
        (r, c) = self.find_empty()
        moved_tile_puzzle = copy.deepcopy(self)
        # Move empty tile upward
        if (direction == "up"):
            if (r-1 >= 0):
                moved_tile_puzzle.layout[r][c],
moved_tile_puzzle.layout[r-1][c] = moved_tile_puzzle.layout[r-1][c],
moved_tile_puzzle.layout[r][c]
                return moved_tile_puzzle
            else:
                return None
        # Move empty tile downward
        elif (direction == "down"):
            if (r+1 < self.n):
                moved_tile_puzzle.layout[r][c],
moved_tile_puzzle.layout[r+1][c] = moved_tile_puzzle.layout[r+1][c],
moved_tile_puzzle.layout[r][c]
                return moved_tile_puzzle
            else:
                return None
        # Move empty tile leftward
        elif (direction == "left"):
            if (c-1 >= 0):
                moved_tile_puzzle.layout[r][c],
moved_tile_puzzle.layout[r][c-1] = moved_tile_puzzle.layout[r][c-1],
moved_tile_puzzle.layout[r][c]
                return moved_tile_puzzle
            else:
                return None
        # Move empty tile rightward
        elif (direction == "right"):
            if (c+1 < self.n):
                moved_tile_puzzle.layout[r][c],
moved_tile_puzzle.layout[r][c+1] = moved_tile_puzzle.layout[r][c+1],
moved_tile_puzzle.layout[r][c]
                return moved_tile_puzzle
            else:
                return None
        else:
            return None

    # Check whether puzzle is solvable or not
    def is_solvable(self):
        (r, c) = self.find_empty()

        flattened_layout = [num for arr in self.layout for num in arr]

        x = (r+c) % 2
        print(f"X = {x}")

        sum = 0
        for i in range(len(flattened_layout)):
            inversion = 0
            for j in range(i+1, len(flattened_layout)):
```



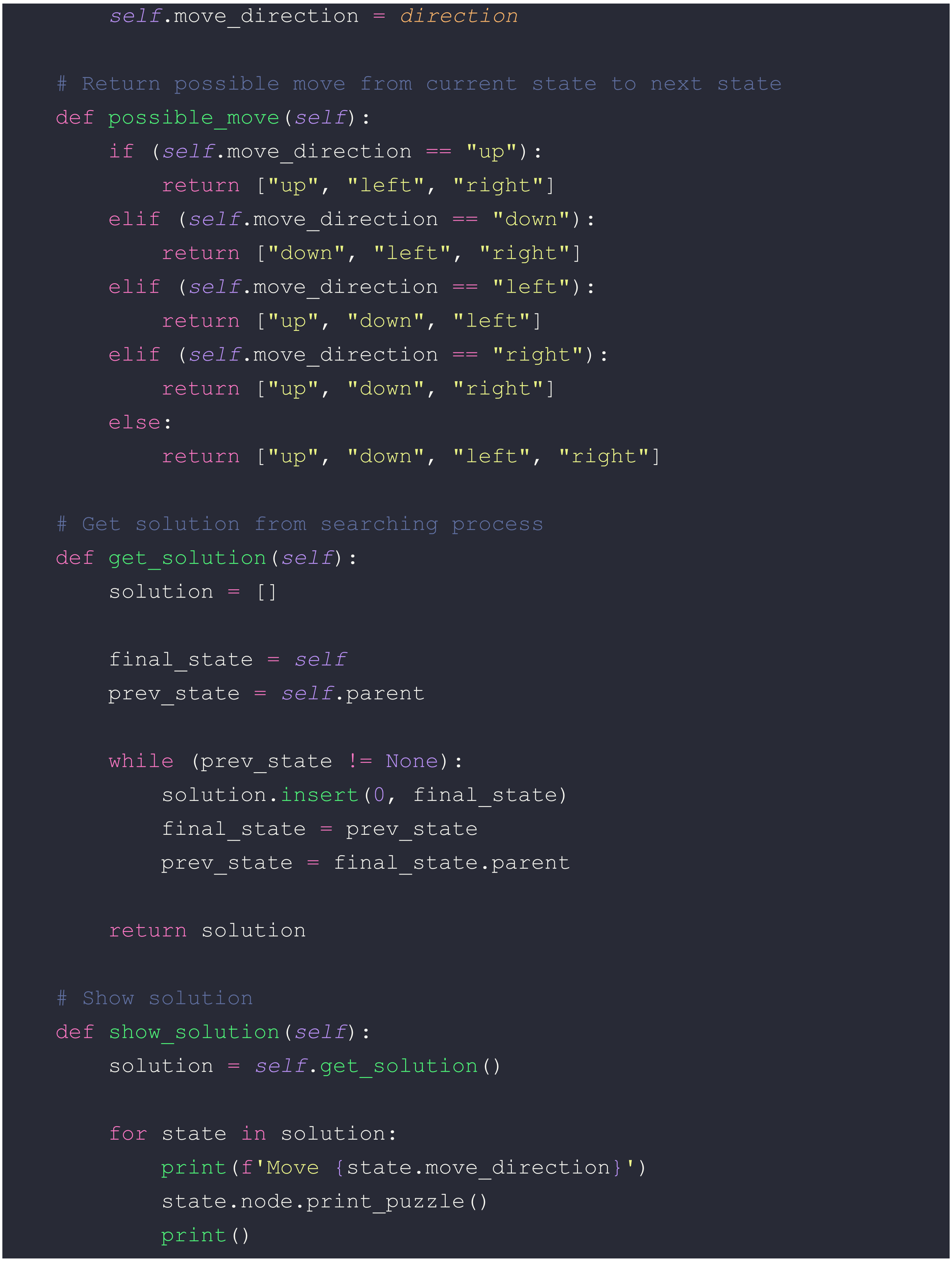
```
                if (flattened_layout[i] > flattened_layout[j]):
                    inversion += 1
            sum += inversion
            print(f"Inversion ({i}) = {inversion}")

        is_even = (sum + x) % 2 == 0
        print(f"Total inversions = {sum}")
        print(f"Total inversions + X = {sum + x} %s" % ("(even)" if
(is_even) else "(odd)"))

        return is_even

    # Print puzzle layout
    def print_puzzle(self):
        for row in self.layout:
            print("-----------------")
            print("|", end="")
            for num in row:
                print("%2s" % (num if num != 16 else " "), end=" ")
                print("|", end="")
            print()
        print("-----------------")

    # Calculate total misplaced tiles
    def calculate_misplaced_tiles(self):
        total = 0
        flattened_layout = [num for arr in self.layout for num in arr]

        for i in range(len(flattened_layout)):
            if (flattened_layout[i] != i+1):
                total += 1

        return total

    # Check if the puzzle is solved
    def is_solved(self):
        flattened_layout = [num for arr in self.layout for num in arr]

        for i in range(len(flattened_layout)):
            if (flattened_layout[i] != i+1):
                return False

        return True
```

3. prioqueue.py

```
from statespacetree import StateSpaceTree

class PriorityQueue:
    def __init__(self):
        self.queue = []

    # Check emptiness
    def is_empty(self):
        return len(self.queue) == 0

    # Enqueue object to queue
    def enqueue(self, object):
        i = 0
        flag = False

        while (not flag and i < len(self.queue)):
            if (object.depth + object.node.calculate_misplaced_tiles() <= self.queue[i].depth + self.queue[i].node.calculate_misplaced_tiles()):
                flag = True
            else:
                i += 1

        self.queue.insert(i, object)

    # Dequeue queue
    def dequeue(self):
        current = self.queue[0]
        self.queue.pop(0)
        return current

    # Solving 15-puzzle
    def solve(self):
        # Solving puzzle
        while (not self.is_empty()):
            # Processing current state
            current_state = self.dequeue()

            # Check if the puzzle is solved
            if (current_state.node.is_solved()):
                final_state = current_state
                break

            # List of possible move directions
            possible_move_directions = current_state.possible_move()

            # Start searching for solution
            for direction in possible_move_directions:
                # Generate node
                new_node =
StateSpaceTree(current_state.node.move(direction))
                new_node.add_depth(current_state)
                new_node.set_parent(current_state)
                new_node.set_move_direction(direction)

                # Check if move process is succeed
                if (new_node.node != None):
                    self.enqueue(new_node)

        return final_state
```

4. statespacetree.py

```
class StateSpaceTree:
    node_generated = 0

    # Constructor
    def __init__(self, node):
        self.node = node
        self.depth = 0
        self.parent = None
        self.move_direction = ""
        if (node != None):
            self.__class__.node_generated += 1

    # Add depth of state space tree
    def add_depth(self, parent):
        self.depth = parent.depth + 1

    # Set parent of state space tree
    def set_parent(self, parent):
        self.parent = parent

    # Set move direction
    def set_move_direction(self, direction):
```



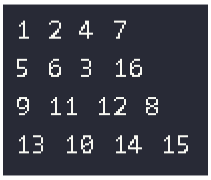
```python
        self.move_direction = direction

    # Return possible move from current state to next state
    def possible_move(self):
        if (self.move_direction == "up"):
            return ["up", "left", "right"]
        elif (self.move_direction == "down"):
            return ["down", "left", "right"]
        elif (self.move_direction == "left"):
            return ["up", "down", "left"]
        elif (self.move_direction == "right"):
            return ["up", "down", "right"]
        else:
            return ["up", "down", "left", "right"]

    # Get solution from searching process
    def get_solution(self):
        solution = []

        final_state = self
        prev_state = self.parent

        while (prev_state != None):
            solution.insert(0, final_state)
            final_state = prev_state
            prev_state = final_state.parent

        return solution

    # Show solution
    def show_solution(self):
        solution = self.get_solution()

        for state in solution:
            print(f'Move {state.move_direction}')
            state.node.print_puzzle()
            print()
```

Screenshot Input dan *Output*

Berikut adalah *screenshot input* dan *output* dari program ini:

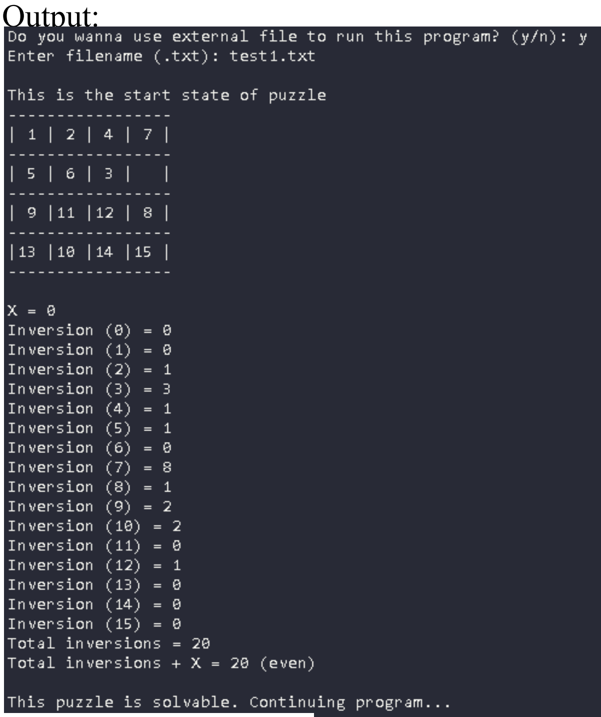
1. test1.txt

   Input:

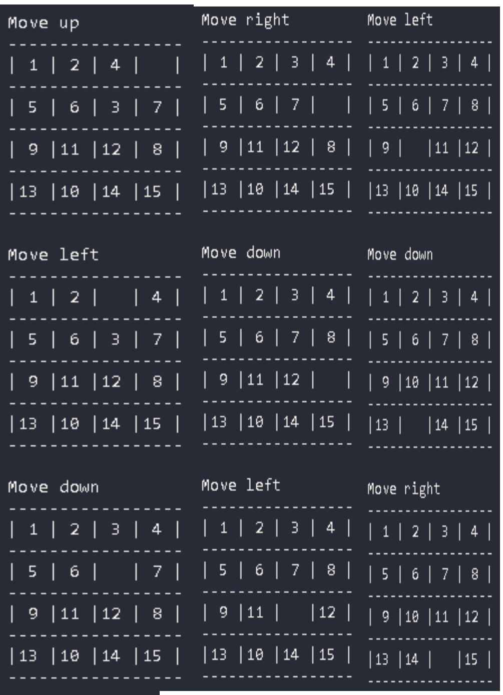


```
1 2 4 7
5 6 3 16
9 11 12 8
13 10 14 15
```

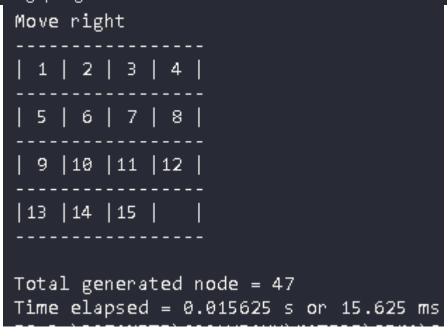
   Output:



```
Do you wanna use external file to run this program? (y/n): y
Enter filename (.txt): test1.txt

This is the start state of puzzle
-----------------
| 1 | 2 | 4 | 7 |
-----------------
| 5 | 6 | 3 |   |
-----------------
| 9 |11 |12 | 8 |
-----------------
|13 |10 |14 |15 |
-----------------

X = 0
Inversion (0) = 0
Inversion (1) = 0
Inversion (2) = 1
Inversion (3) = 3
Inversion (4) = 1
Inversion (5) = 1
Inversion (6) = 0
Inversion (7) = 8
Inversion (8) = 1
Inversion (9) = 2
Inversion (10) = 2
Inversion (11) = 0
Inversion (12) = 1
Inversion (13) = 0
Inversion (14) = 0
Inversion (15) = 0
Total inversions = 20
Total inversions + X = 20 (even)

This puzzle is solvable. Continuing program...
```



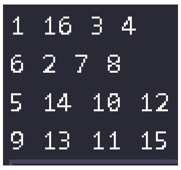
```
Move up
-----------------
| 1 | 2 | 4 |   |
-----------------
| 5 | 6 | 3 | 7 |
-----------------
| 9 |11 |12 | 8 |
-----------------
|13 |10 |14 |15 |
-----------------

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

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

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

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

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

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

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

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



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

Total generated node = 47
Time elapsed = 0.015625 s or 15.625 ms
```

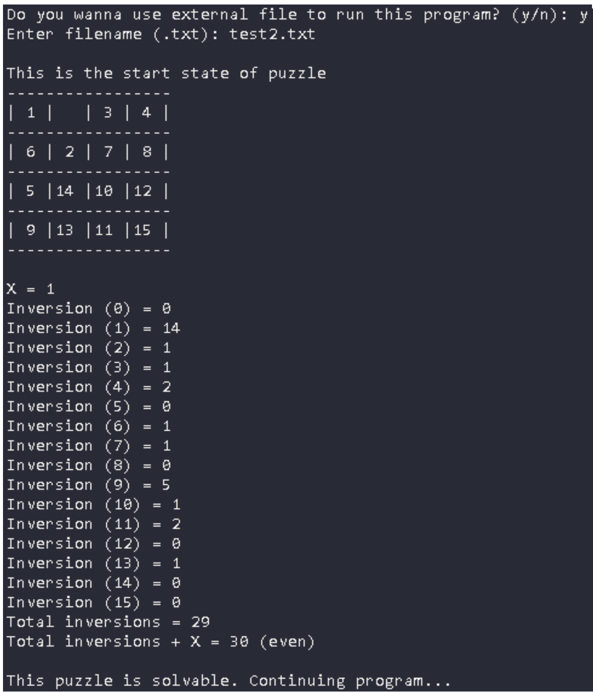
2. test2.txt

   Input:

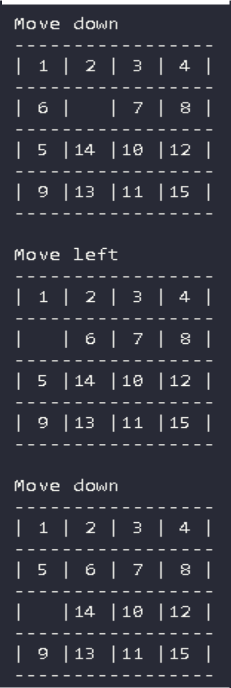


```
1 16 3 4
6 2 7 8
5 14 10 12
9 13 11 15
```

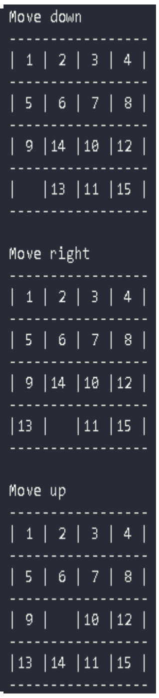
Output:



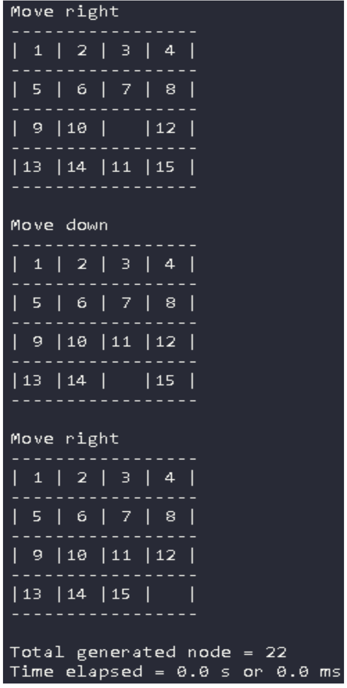
```
Do you wanna use external file to run this program? (y/n): y
Enter filename (.txt): test2.txt

This is the start state of puzzle
-----------------
| 1 |   | 3 | 4 |
-----------------
| 6 | 2 | 7 | 8 |
-----------------
| 5 |14 |10 |12 |
-----------------
| 9 |13 |11 |15 |
-----------------

X = 1
Inversion (0) = 0
Inversion (1) = 14
Inversion (2) = 1
Inversion (3) = 1
Inversion (4) = 2
Inversion (5) = 0
Inversion (6) = 1
Inversion (7) = 1
Inversion (8) = 0
Inversion (9) = 5
Inversion (10) = 1
Inversion (11) = 2
Inversion (12) = 0
Inversion (13) = 1
Inversion (14) = 0
Inversion (15) = 0
Total inversions = 29
Total inversions + X = 30 (even)

This puzzle is solvable. Continuing program...
```



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

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

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



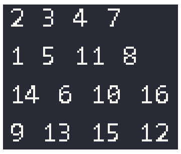
```
Move down
-----------------
| 1 | 2 | 3 | 4 |
-----------------
| 5 | 6 | 7 | 8 |
-----------------
| 9 |14 |10 |12 |
-----------------
|   |13 |11 |15 |
-----------------

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

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



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

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

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

Total generated node = 22
Time elapsed = 0.0 s or 0.0 ms
```

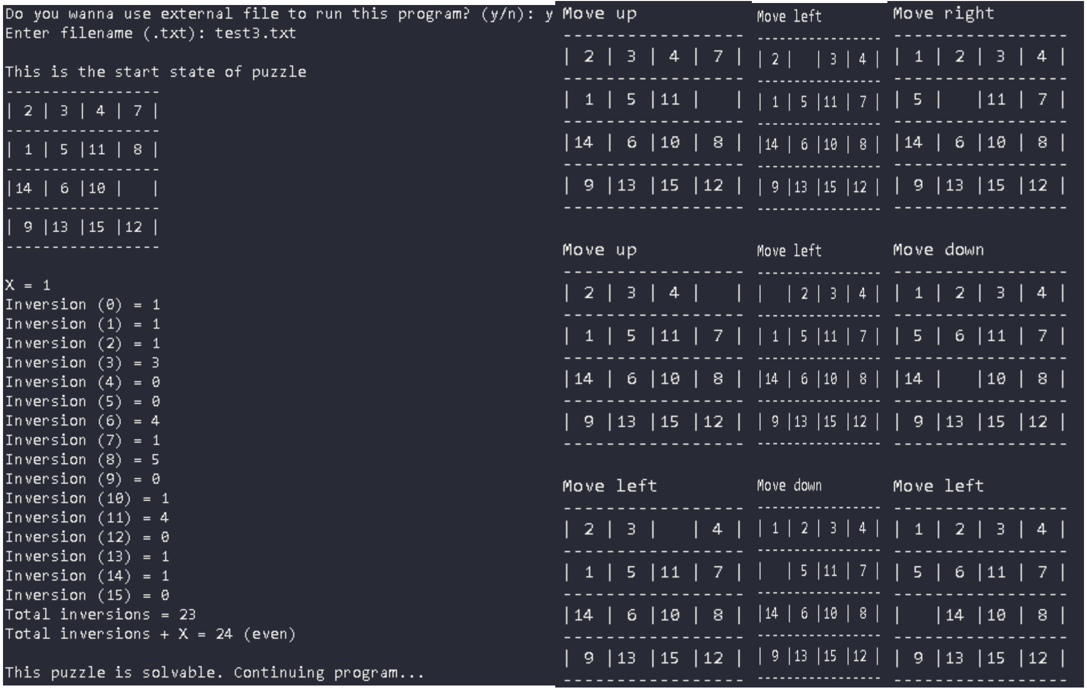
3. test3.txt

Input:



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

Output:

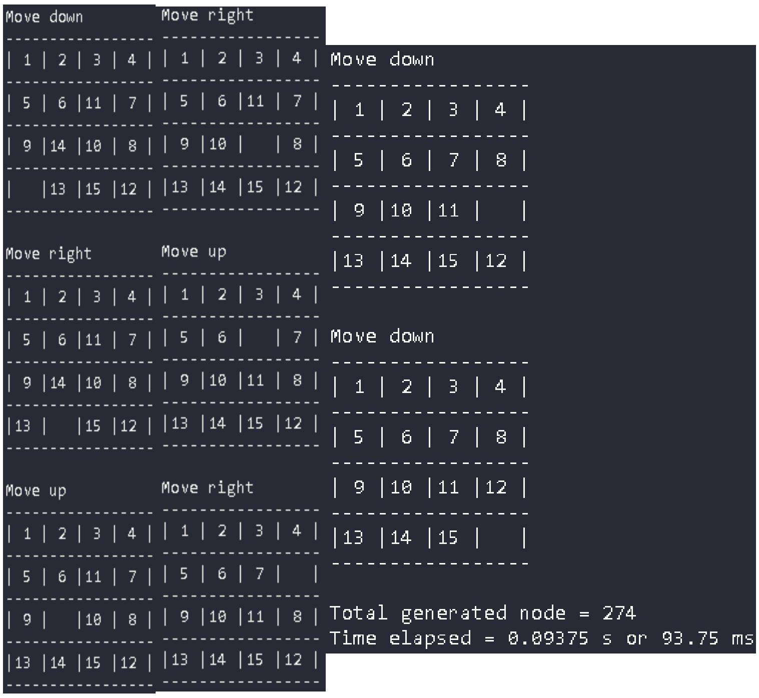


```
Do you wanna use external file to run this program? (y/n): y
Enter filename (.txt): test3.txt

This is the start state of puzzle
----------------
| 2 | 3 | 4 | 7 |
----------------
| 1 | 5 |11 | 8 |
----------------
|14 | 6 |10 |   |
----------------
| 9 |13 |15 |12 |
----------------

X = 1
Inversion (0) = 1
Inversion (1) = 1
Inversion (2) = 1
Inversion (3) = 3
Inversion (4) = 0
Inversion (5) = 0
Inversion (6) = 4
Inversion (7) = 1
Inversion (8) = 5
Inversion (9) = 0
Inversion (10) = 1
Inversion (11) = 4
Inversion (12) = 0
Inversion (13) = 1
Inversion (14) = 1
Inversion (15) = 0
Total inversions = 23
Total inversions + X = 24 (even)

This puzzle is solvable. Continuing program...
```

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

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

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

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

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

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

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

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

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



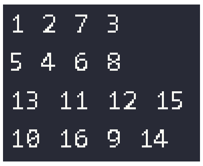
```
Move down
-----------------
| 1 | 2 | 3 | 4 |
-----------------
| 5 | 6 |11 | 7 |
-----------------
| 9 |14 |10 | 8 |
-----------------
|   |13 |15 |12 |
-----------------

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

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

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

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

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

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

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

Total generated node = 274
Time elapsed = 0.09375 s or 93.75 ms
```

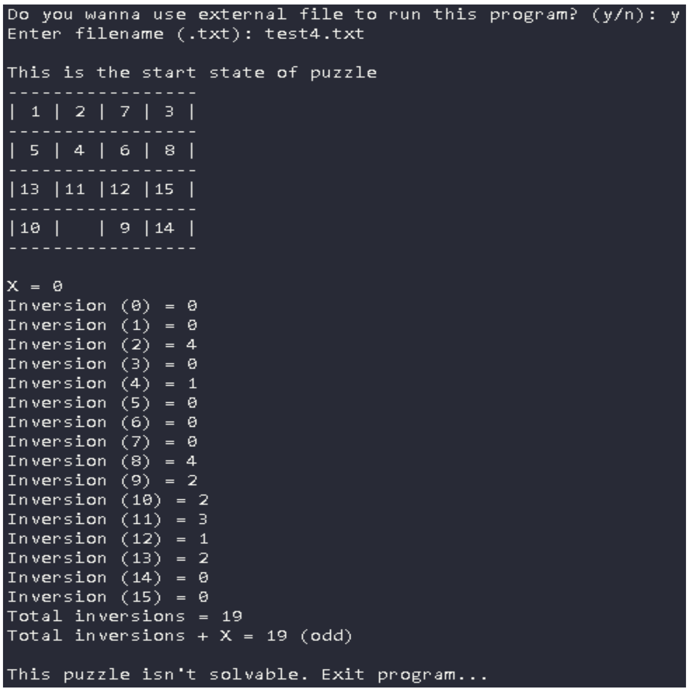
4. test4.txt

Input:



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

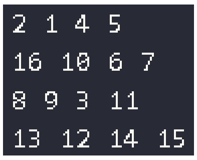
Output:



```
Do you wanna use external file to run this program? (y/n): y
Enter filename (.txt): test4.txt

This is the start state of puzzle
-----------------
| 1 | 2 | 7 | 3 |
-----------------
| 5 | 4 | 6 | 8 |
-----------------
|13 |11 |12 |15 |
-----------------
|10 |   | 9 |14 |
-----------------

X = 0
Inversion (0) = 0
Inversion (1) = 0
Inversion (2) = 4
Inversion (3) = 0
Inversion (4) = 1
Inversion (5) = 0
Inversion (6) = 0
Inversion (7) = 0
Inversion (8) = 4
Inversion (9) = 2
Inversion (10) = 2
Inversion (11) = 3
Inversion (12) = 1
Inversion (13) = 2
Inversion (14) = 0
Inversion (15) = 0
Total inversions = 19
Total inversions + X = 19 (odd)

This puzzle isn't solvable. Exit program...
```

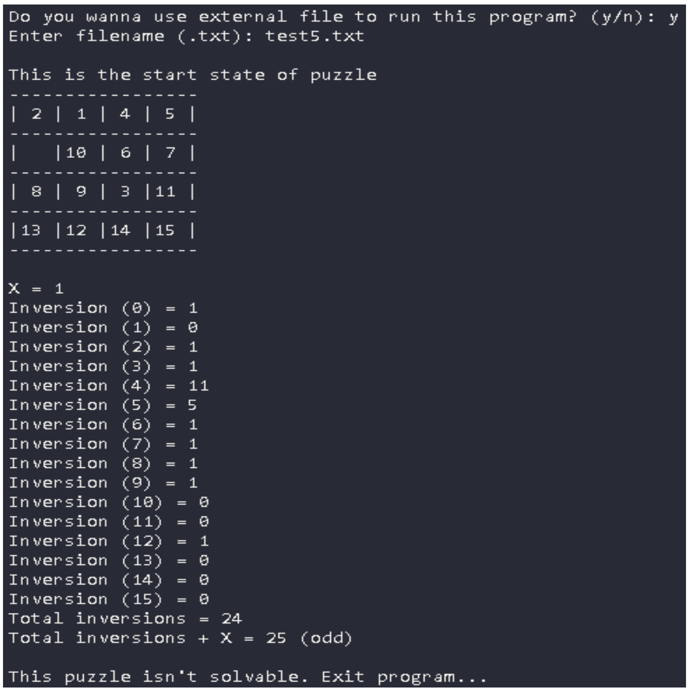
5. test5.txt

Input:



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

Output:



```
Do you wanna use external file to run this program? (y/n): y
Enter filename (.txt): test5.txt

This is the start state of puzzle
-----------------
| 2 | 1 | 4 | 5 |
-----------------
|   |10 | 6 | 7 |
-----------------
| 8 | 9 | 3 |11 |
-----------------
|13 |12 |14 |15 |
-----------------

X = 1
Inversion (0) = 1
Inversion (1) = 0
Inversion (2) = 1
Inversion (3) = 1
Inversion (4) = 11
Inversion (5) = 5
Inversion (6) = 1
Inversion (7) = 1
Inversion (8) = 1
Inversion (9) = 1
Inversion (10) = 0
Inversion (11) = 0
Inversion (12) = 1
Inversion (13) = 0
Inversion (14) = 0
Inversion (15) = 0
Total inversions = 24
Total inversions + X = 25 (odd)

This puzzle isn't solvable. Exit program...
```

Repository Github

Berikut adalah *link repository* Github yang akan diatur ke *public* setelah pengumpulan: https://github.com/OjaanIr/TucilStima-15Puzzle

Status Program

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		✓