Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

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

Disusun oleh:

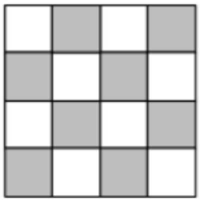
Muhammad Akyas David Al Aleey 13520011

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

A. Algoritma *Branch and Bound* untuk Menyelesaikan Persoalan 15-Puzzle

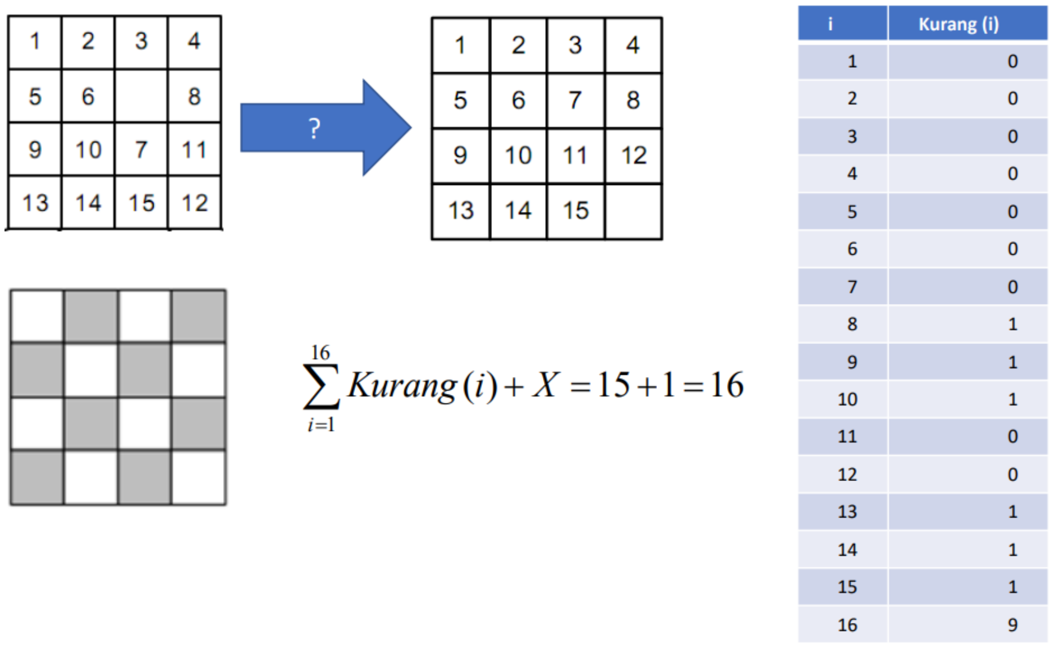
Untuk menyelesaikan persoalan pada tugas kecil ini, digunakan algoritma *branch and bound* yang akan menjabarkan *state* dari suatu persoalan bila diketahui *state* tujuan dari persoalan tersebut. Algoritma *branch and bound* dapat mengoptimalisasi persoalan tersebut karena algoritma tersebut pada dasarnya akan meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan persoalan.

Implementasi algoritma branch and bound diawali dengan menerima sebuah input berupa matriks berdimensi 4x4 dengan setiap elemennya bernilai unik dari 0 hingga 15. Input ini dapat berupa masukan dari pengguna maupun masukan file. Untuk menentukan apakah *puzzle* yang dimasukkan dapat diselesaikan, akan dihitung terlebih dahulu nilai $\sum kurang(i) + X$ dari *puzzle* tersebut. Jika hasilnya berupa bilangan genap, maka persoalan *puzzle* tersebut dapat diselesaikan. Sebaliknya, jika hasilnya berupa bilangan ganjil, maka persoalan *puzzle* tersebut tidak dapat diselesaikan. Nilai X bernilai 1 jika posisi awal sel yang kosong atau bernilai 0 berada pada sel yang diarsir dan bernilai nol jika terletak pada sel yang tidak diarsir.



Gambar 1.1 Penentuan sel yang diarsir

Nilai $kurang(i)$ merupakan banyaknya ubin bernomor j sedemikian sehingga $j < i$ dan $posisi(j) > posisi(i)$. $posisi(i)$ merupakan posisi ubin i pada susunan *puzzle* yang diperiksa. Berikut contoh ilustrasi perhitungan nilai $\sum kurang(i) + X$.



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

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

$$\sum_{i=1}^{16} Kurang(i) + X = 15 + 1 = 16$$

i	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
16	9

Gambar 1.2 Ilustrasi perhitungan nilai $\sum kurang(i) + X$

Seperti yang terlihat di atas, hasil penjumlahan seluruh kurang(i) dengan X menghasilkan nilai genap. Oleh karena itu, langkah penyelesaian persoalan akan berlanjut dengan menggunakan algoritma *branch and bound*. Langkah-langkah yang dilakukan yakni:

1. Masukkan simpul akar ke dalam suatu priority queue. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan.
2. Jika priority queue kosong, maka Stop.
3. Jika priority queue tidak kosong, maka akan dipilih elemen pertama dari priority queue. Setiap simpul di dalam priority queue diurutkan secara menaik dengan aturan pengurutan simpul elemen yaitu mempertimbangkan nilai *cost* setiap simpulnya. Cost tersebut didapatkan dari persamaan berikut:

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

   dengan,

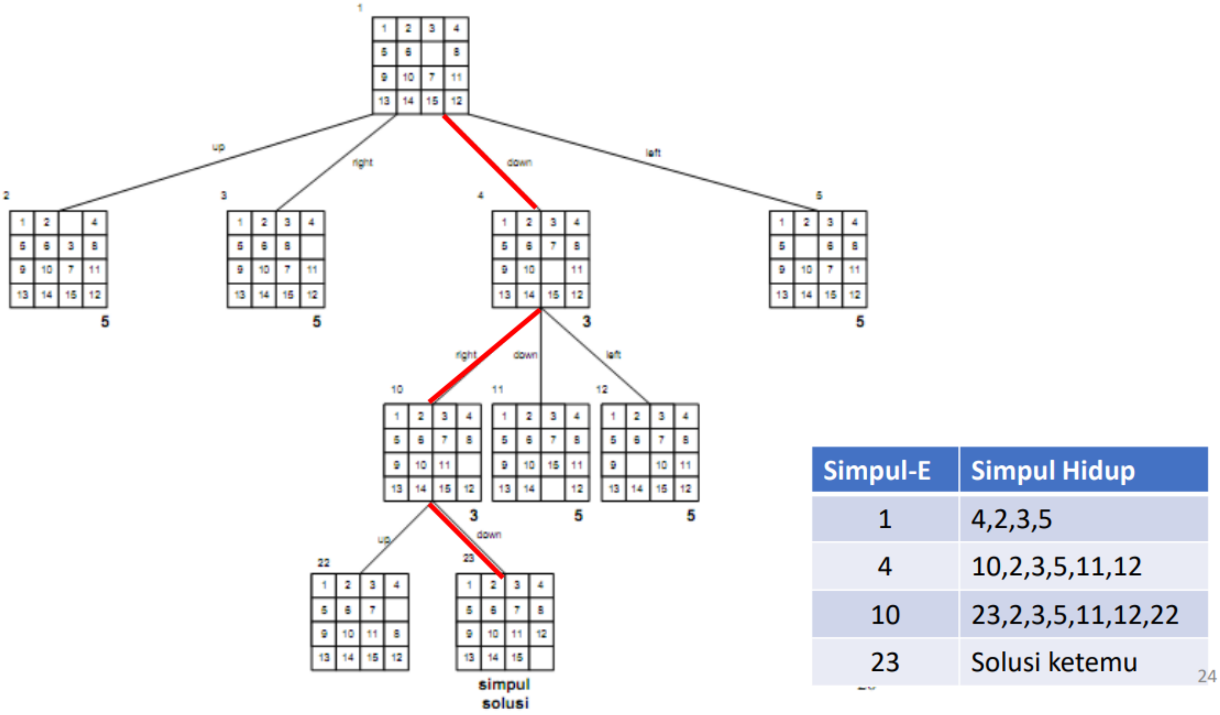
   $$c(i) = ongkos\ untuk\ simpul\ i$$

   $$f(i) = panjang\ lintasan\ dari\ simpul\ akar\ ke\ i$$

   $$g(i) = jumlah\ ubin\ tidak\ kosong\ yang\ tidak\ terdapat\ pada\ susunan\ terakhir$$
4. Jika simpul i adalah simpul solusi, yakni $g(i)$ bernilai 0, berarti solusi sudah ditemukan.
5. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.

6. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
7. Kembali ke langkah 2.

Berikut ilustrasi pembentukan pohon ruang status 15-Puzzle dengan algoritma *branch and bound*.

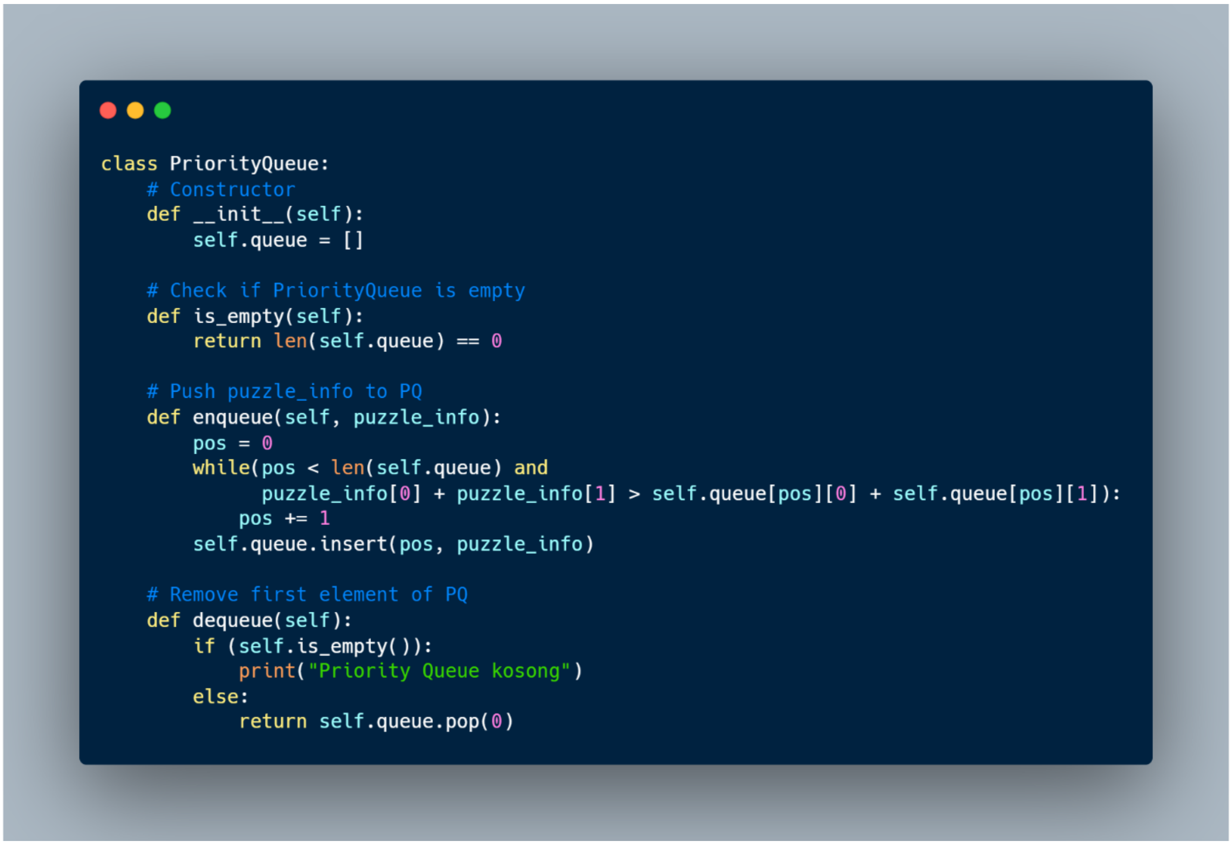


Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

Gambar 1.3 Ilustrasi pohon ruang status 15-Puzzle

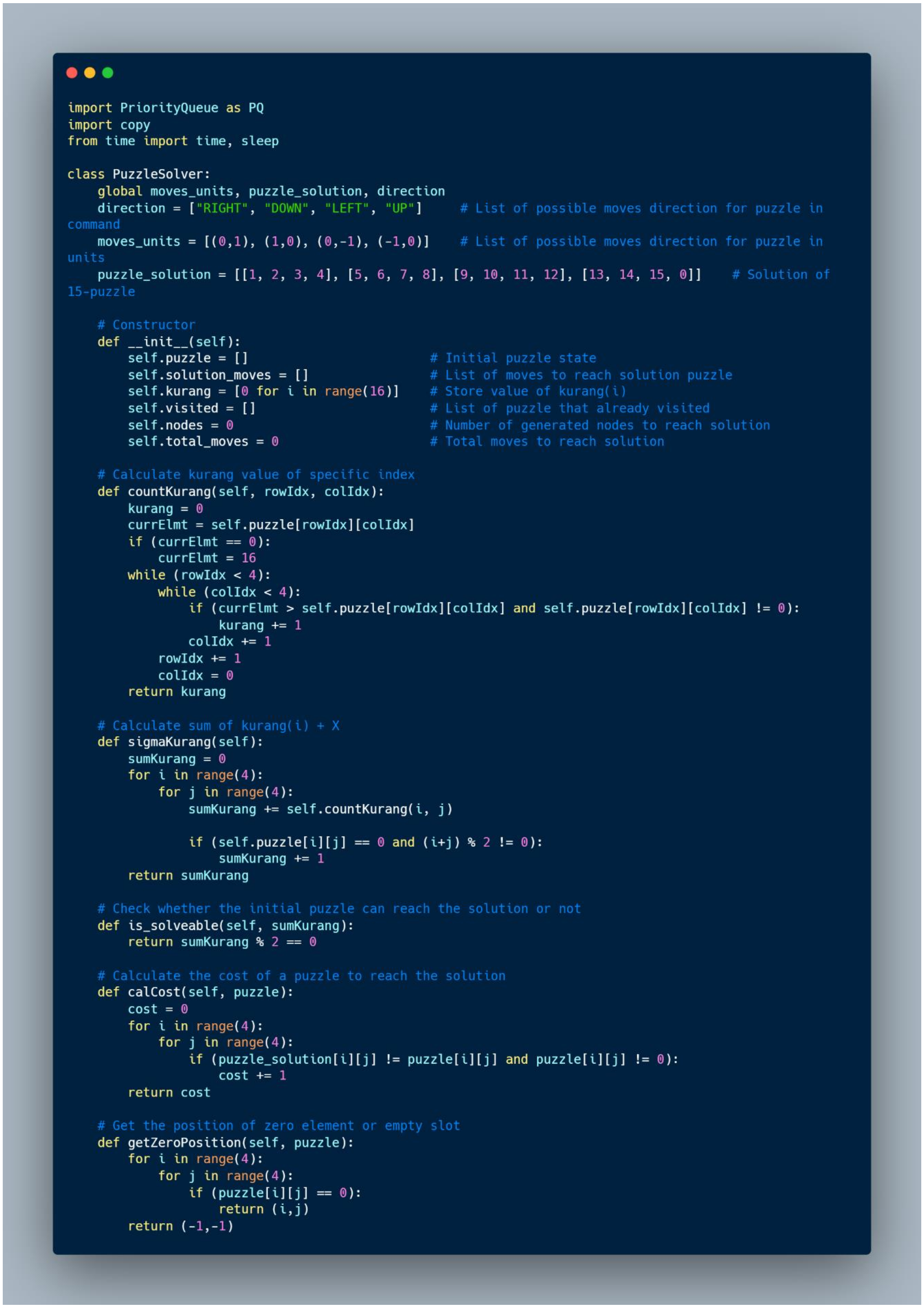
B. *Source Code* Program

1. PriorityQueue.py

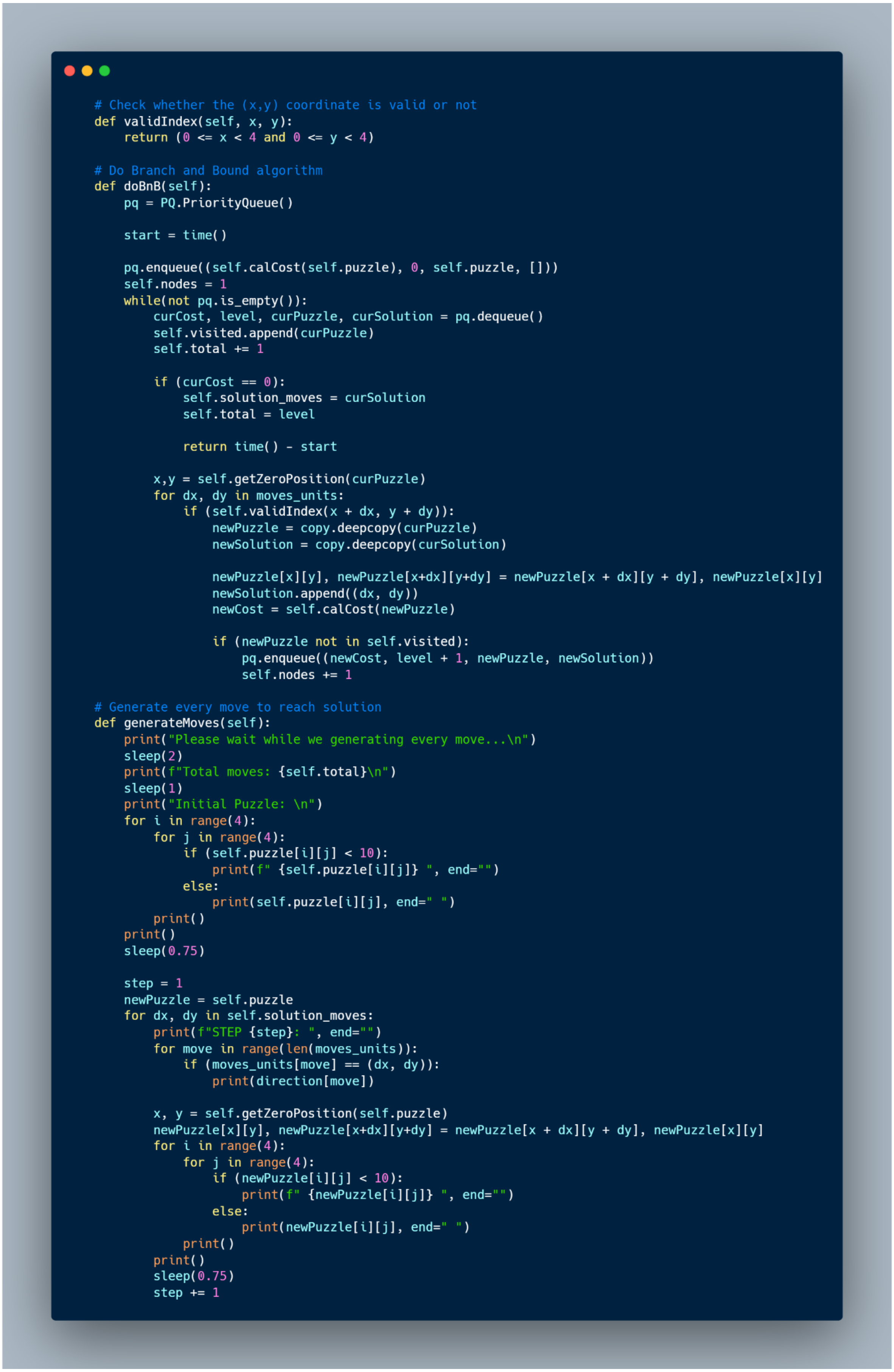


```python
class PriorityQueue:
    # Constructor
    def __init__(self):
        self.queue = []

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

    # Push puzzle_info to PQ
    def enqueue(self, puzzle_info):
        pos = 0
        while(pos < len(self.queue) and
              puzzle_info[0] + puzzle_info[1] > self.queue[pos][0] + self.queue[pos][1]):
            pos += 1
        self.queue.insert(pos, puzzle_info)

    # Remove first element of PQ
    def dequeue(self):
        if (self.is_empty()):
            print("Priority Queue kosong")
        else:
            return self.queue.pop(0)
```

2. PuzzleSolver.py

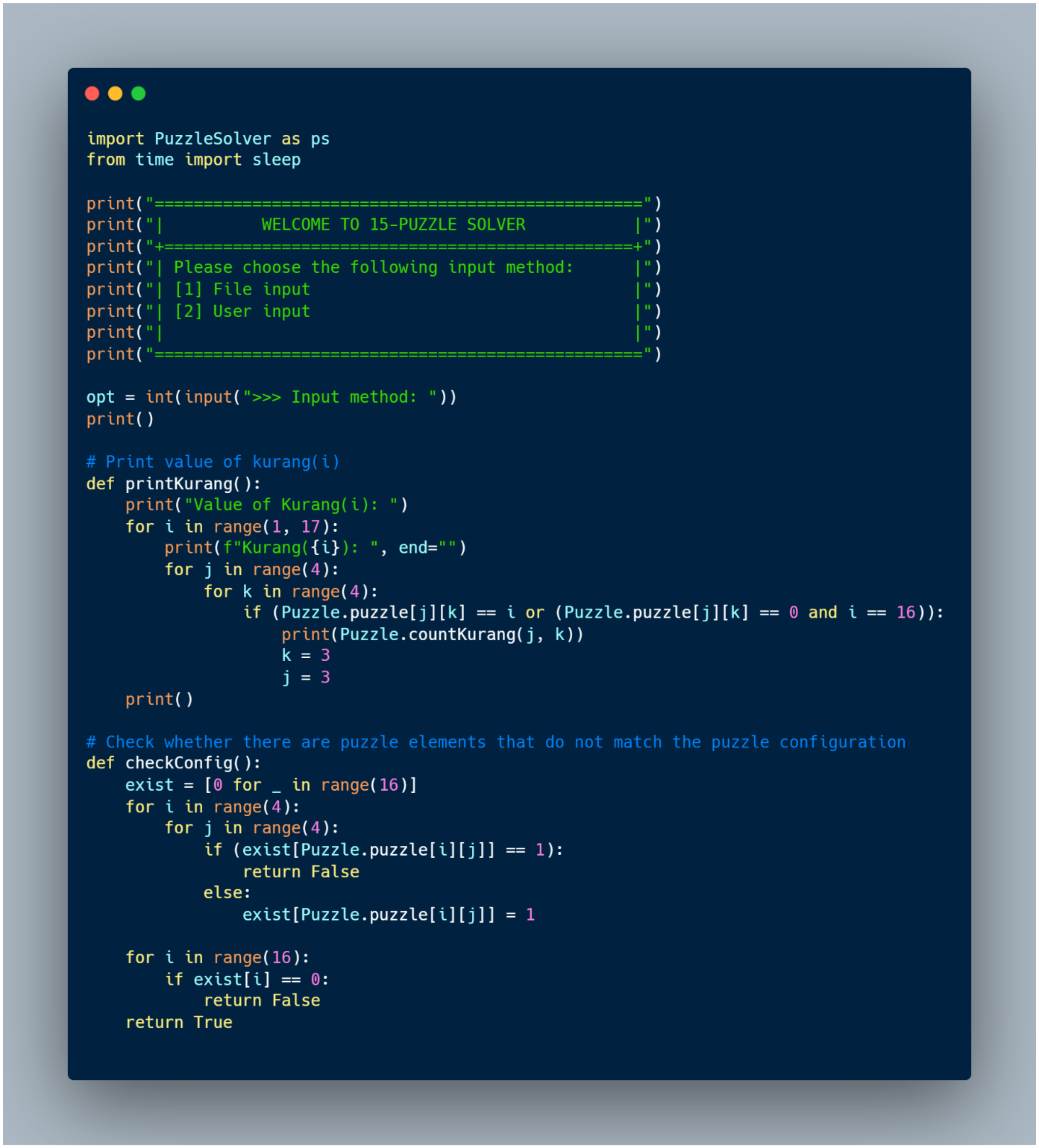


```
import PriorityQueue as PQ
import copy
from time import time, sleep

class PuzzleSolver:
    global moves_units, puzzle_solution, direction
    direction = ["RIGHT", "DOWN", "LEFT", "UP"]     # List of possible moves direction for puzzle in
command
    moves_units = [(0,1), (1,0), (0,-1), (-1,0)]    # List of possible moves direction for puzzle in
units
    puzzle_solution = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]    # Solution of
15-puzzle

    # Constructor
    def __init__(self):
        self.puzzle = []                           # Initial puzzle state
        self.solution_moves = []                   # List of moves to reach solution puzzle
        self.kurang = [0 for i in range(16)]       # Store value of kurang(i)
        self.visited = []                          # List of puzzle that already visited
        self.nodes = 0                             # Number of generated nodes to reach solution
        self.total_moves = 0                       # Total moves to reach solution

    # Calculate kurang value of specific index
    def countKurang(self, rowIdx, colIdx):
        kurang = 0
        currElmt = self.puzzle[rowIdx][colIdx]
        if (currElmt == 0):
            currElmt = 16
        while (rowIdx < 4):
            while (colIdx < 4):
                if (currElmt > self.puzzle[rowIdx][colIdx] and self.puzzle[rowIdx][colIdx] != 0):
                    kurang += 1
                colIdx += 1
            rowIdx += 1
            colIdx = 0
        return kurang

    # Calculate sum of kurang(i) + X
    def sigmaKurang(self):
        sumKurang = 0
        for i in range(4):
            for j in range(4):
                sumKurang += self.countKurang(i, j)

                if (self.puzzle[i][j] == 0 and (i+j) % 2 != 0):
                    sumKurang += 1
        return sumKurang

    # Check whether the initial puzzle can reach the solution or not
    def is_solveable(self, sumKurang):
        return sumKurang % 2 == 0

    # Calculate the cost of a puzzle to reach the solution
    def calCost(self, puzzle):
        cost = 0
        for i in range(4):
            for j in range(4):
                if (puzzle_solution[i][j] != puzzle[i][j] and puzzle[i][j] != 0):
                    cost += 1
        return cost

    # Get the position of zero element or empty slot
    def getZeroPosition(self, puzzle):
        for i in range(4):
            for j in range(4):
                if (puzzle[i][j] == 0):
                    return (i,j)
        return (-1,-1)
```

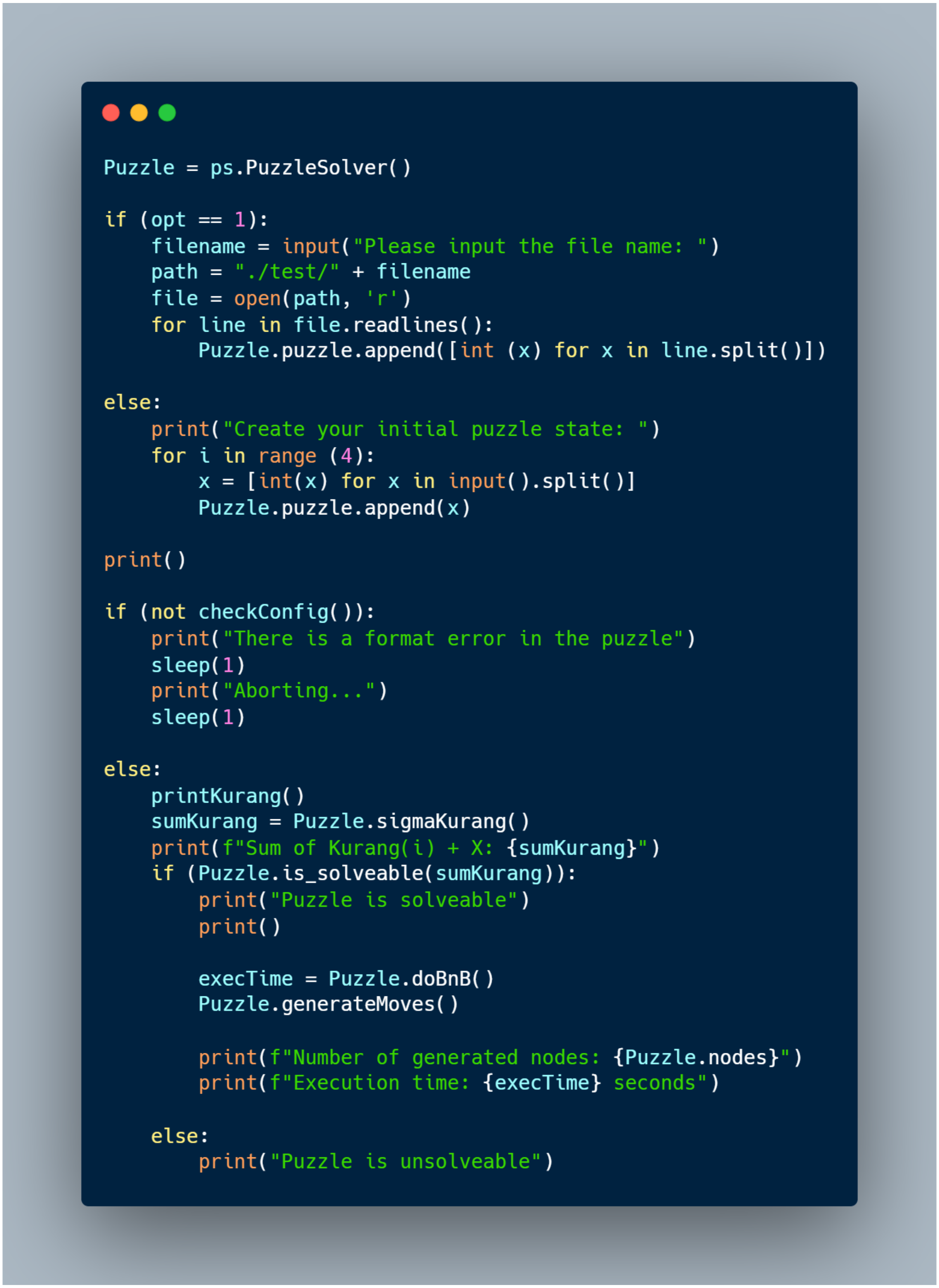


```python
    # Check whether the (x,y) coordinate is valid or not
    def validIndex(self, x, y):
        return (0 <= x < 4 and 0 <= y < 4)

    # Do Branch and Bound algorithm
    def doBnB(self):
        pq = PQ.PriorityQueue()

        start = time()

        pq.enqueue((self.calCost(self.puzzle), 0, self.puzzle, []))
        self.nodes = 1
        while(not pq.is_empty()):
            curCost, level, curPuzzle, curSolution = pq.dequeue()
            self.visited.append(curPuzzle)
            self.total += 1

            if (curCost == 0):
                self.solution_moves = curSolution
                self.total = level

                return time() - start

            x,y = self.getZeroPosition(curPuzzle)
            for dx, dy in moves_units:
                if (self.validIndex(x + dx, y + dy)):
                    newPuzzle = copy.deepcopy(curPuzzle)
                    newSolution = copy.deepcopy(curSolution)

                    newPuzzle[x][y], newPuzzle[x+dx][y+dy] = newPuzzle[x + dx][y + dy], newPuzzle[x][y]
                    newSolution.append((dx, dy))
                    newCost = self.calCost(newPuzzle)

                    if (newPuzzle not in self.visited):
                        pq.enqueue((newCost, level + 1, newPuzzle, newSolution))
                        self.nodes += 1

    # Generate every move to reach solution
    def generateMoves(self):
        print("Please wait while we generating every move...\n")
        sleep(2)
        print(f"Total moves: {self.total}\n")
        sleep(1)
        print("Initial Puzzle: \n")
        for i in range(4):
            for j in range(4):
                if (self.puzzle[i][j] < 10):
                    print(f" {self.puzzle[i][j]} ", end="")
                else:
                    print(self.puzzle[i][j], end=" ")
            print()
        print()
        sleep(0.75)

        step = 1
        newPuzzle = self.puzzle
        for dx, dy in self.solution_moves:
            print(f"STEP {step}: ", end="")
            for move in range(len(moves_units)):
                if (moves_units[move] == (dx, dy)):
                    print(direction[move])

            x, y = self.getZeroPosition(self.puzzle)
            newPuzzle[x][y], newPuzzle[x+dx][y+dy] = newPuzzle[x + dx][y + dy], newPuzzle[x][y]
            for i in range(4):
                for j in range(4):
                    if (newPuzzle[i][j] < 10):
                        print(f" {newPuzzle[i][j]} ", end="")
                    else:
                        print(newPuzzle[i][j], end=" ")
                print()
            print()
            sleep(0.75)
            step += 1
```

3. Main.py



```python
import PuzzleSolver as ps
from time import sleep

print("==================================================")
print("|          WELCOME TO 15-PUZZLE SOLVER           |")
print("+================================================+")
print("| Please choose the following input method:      |")
print("| [1] File input                                 |")
print("| [2] User input                                 |")
print("|                                                |")
print("==================================================")

opt = int(input(">>> Input method: "))
print()

# Print value of kurang(i)
def printKurang():
    print("Value of Kurang(i): ")
    for i in range(1, 17):
        print(f"Kurang({i}): ", end="")
        for j in range(4):
            for k in range(4):
                if (Puzzle.puzzle[j][k] == i or (Puzzle.puzzle[j][k] == 0 and i == 16)):
                    print(Puzzle.countKurang(j, k))
                    k = 3
                    j = 3
    print()

# Check whether there are puzzle elements that do not match the puzzle configuration
def checkConfig():
    exist = [0 for _ in range(16)]
    for i in range(4):
        for j in range(4):
            if (exist[Puzzle.puzzle[i][j]] == 1):
                return False
            else:
                exist[Puzzle.puzzle[i][j]] = 1

    for i in range(16):
        if exist[i] == 0:
            return False
    return True
```



```python
Puzzle = ps.PuzzleSolver()

if (opt == 1):
    filename = input("Please input the file name: ")
    path = "./test/" + filename
    file = open(path, 'r')
    for line in file.readlines():
        Puzzle.puzzle.append([int (x) for x in line.split()])

else:
    print("Create your initial puzzle state: ")
    for i in range (4):
        x = [int(x) for x in input().split()]
        Puzzle.puzzle.append(x)

print()

if (not checkConfig()):
    print("There is a format error in the puzzle")
    sleep(1)
    print("Aborting...")
    sleep(1)

else:
    printKurang()
    sumKurang = Puzzle.sigmaKurang()
    print(f"Sum of Kurang(i) + X: {sumKurang}")
    if (Puzzle.is_solveable(sumKurang)):
        print("Puzzle is solveable")
        print()

        execTime = Puzzle.doBnB()
        Puzzle.generateMoves()

        print(f"Number of generated nodes: {Puzzle.nodes}")
        print(f"Execution time: {execTime} seconds")

    else:
        print("Puzzle is unsolveable")
```

C. Contoh instansiasi 5 buah persoalan 15-puzzle

1. solveable_1.txt

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

2. solveable_2.txt

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

3. solveable_3.txt

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

4. unsolveable_1.txt

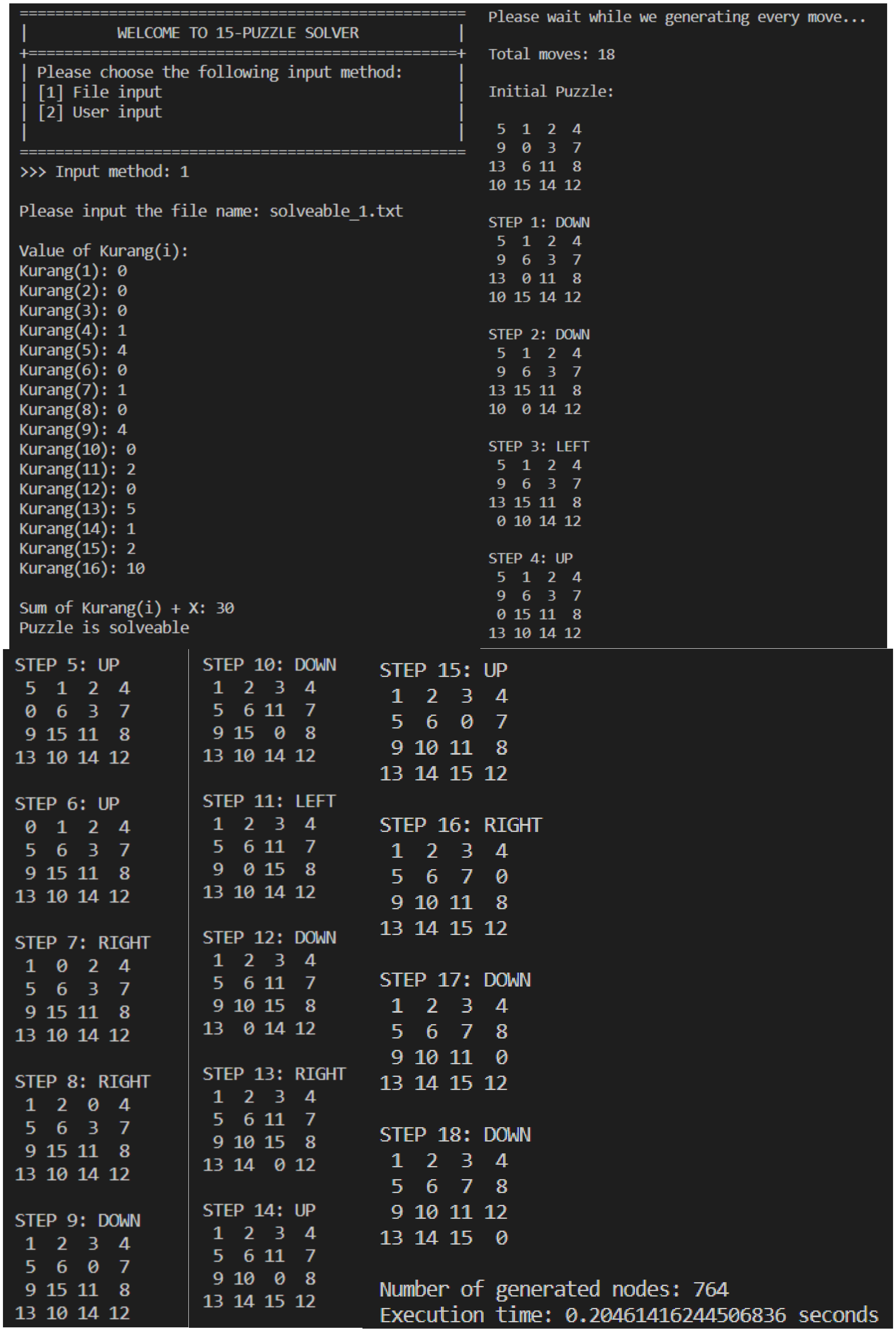
```
5 1 3 4
9 0 13 10
11 15 8 7
6 12 14 2
```

5. unsolveable_2.txt

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

D. *Screenshot* Input dan Output

1. solveable_1.txt



```
================================================
|         WELCOME TO 15-PUZZLE SOLVER          |
+==============================================+
| Please choose the following input method:    |
| [1] File input                               |
| [2] User input                               |
|                                              |
================================================
>>> Input method: 1

Please input the file name: solveable_1.txt

Value of Kurang(i):
Kurang(1): 0
Kurang(2): 0
Kurang(3): 0
Kurang(4): 1
Kurang(5): 4
Kurang(6): 0
Kurang(7): 1
Kurang(8): 0
Kurang(9): 4
Kurang(10): 0
Kurang(11): 2
Kurang(12): 0
Kurang(13): 5
Kurang(14): 1
Kurang(15): 2
Kurang(16): 10

Sum of Kurang(i) + X: 30
Puzzle is solveable
```

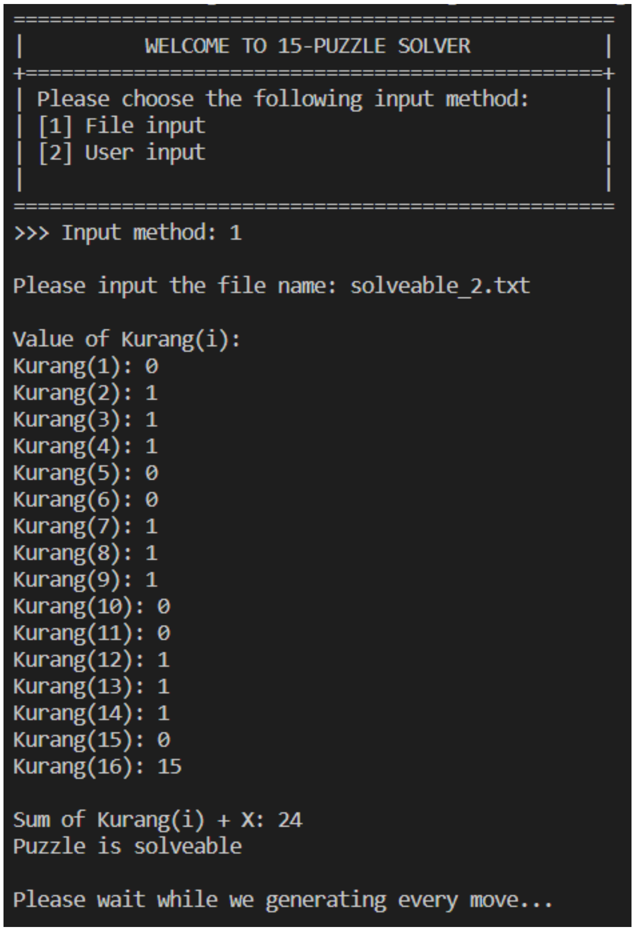
```
Please wait while we generating every move...

Total moves: 18

Initial Puzzle:

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

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

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

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

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

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

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

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

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

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

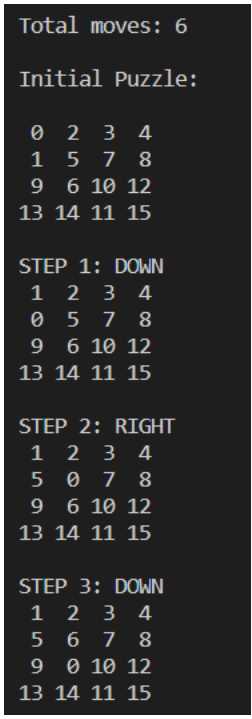
```
STEP 10: DOWN
 1  2  3  4
 5  6 11  7
 9 15  0  8
13 10 14 12

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

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

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

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

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

STEP 16: RIGHT
 1  2  3  4
 5  6  7  0
 9 10 11  8
13 14 15 12

STEP 17: DOWN
 1  2  3  4
 5  6  7  8
 9 10 11  0
13 14 15 12

STEP 18: DOWN
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15  0

Number of generated nodes: 764
Execution time: 0.20461416244506836 seconds
```

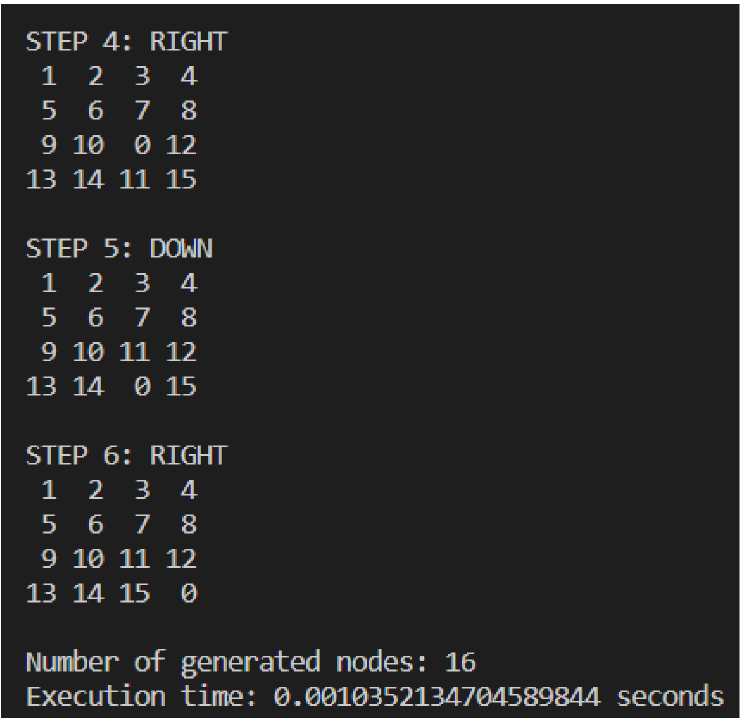
2. solveable_2.txt



```
==================================================
|           WELCOME TO 15-PUZZLE SOLVER          |
+================================================+
| Please choose the following input method:      |
| [1] File input                                 |
| [2] User input                                 |
|                                                |
==================================================
>>> Input method: 1

Please input the file name: solveable_2.txt

Value of Kurang(i):
Kurang(1): 0
Kurang(2): 1
Kurang(3): 1
Kurang(4): 1
Kurang(5): 0
Kurang(6): 0
Kurang(7): 1
Kurang(8): 1
Kurang(9): 1
Kurang(10): 0
Kurang(11): 0
Kurang(12): 1
Kurang(13): 1
Kurang(14): 1
Kurang(15): 0
Kurang(16): 15

Sum of Kurang(i) + X: 24
Puzzle is solveable

Please wait while we generating every move...
```

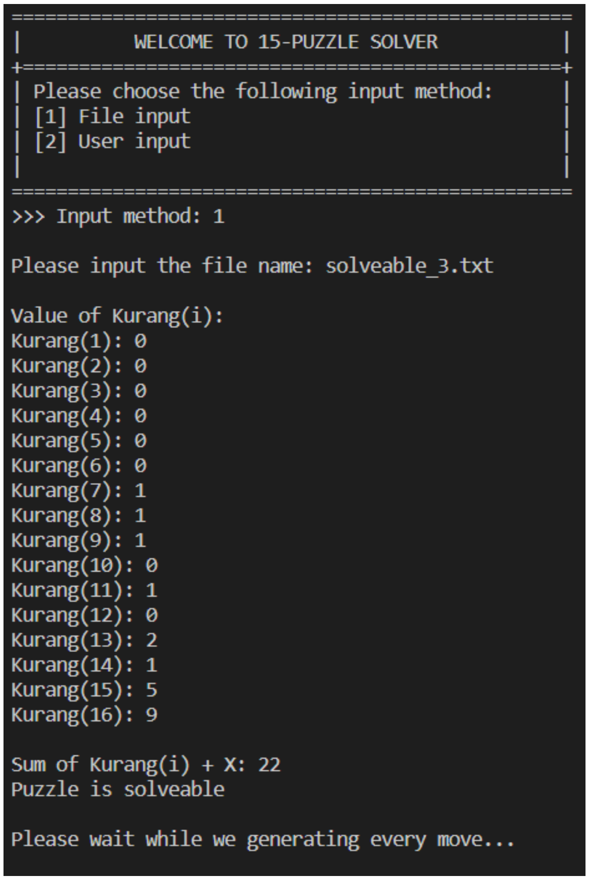


```
Total moves: 6

Initial Puzzle:

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

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

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

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

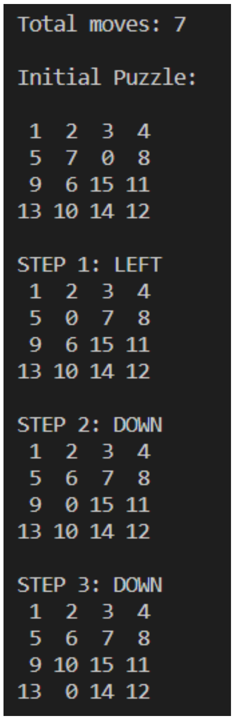


```
STEP 4: RIGHT
 1  2  3  4
 5  6  7  8
 9 10  0 12
13 14 11 15

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

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

Number of generated nodes: 16
Execution time: 0.0010352134704589844 seconds
```

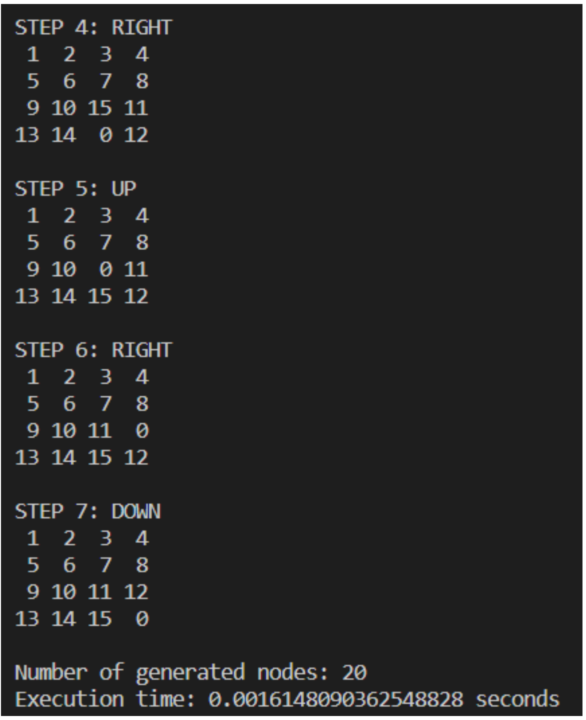
3. solveable_3.txt



```
==================================================
|          WELCOME TO 15-PUZZLE SOLVER           |
+================================================+
| Please choose the following input method:      |
| [1] File input                                 |
| [2] User input                                 |
|                                                |
==================================================
>>> Input method: 1

Please input the file name: solveable_3.txt

Value of Kurang(i):
Kurang(1): 0
Kurang(2): 0
Kurang(3): 0
Kurang(4): 0
Kurang(5): 0
Kurang(6): 0
Kurang(7): 1
Kurang(8): 1
Kurang(9): 1
Kurang(10): 0
Kurang(11): 1
Kurang(12): 0
Kurang(13): 2
Kurang(14): 1
Kurang(15): 5
Kurang(16): 9

Sum of Kurang(i) + X: 22
Puzzle is solveable

Please wait while we generating every move...
```

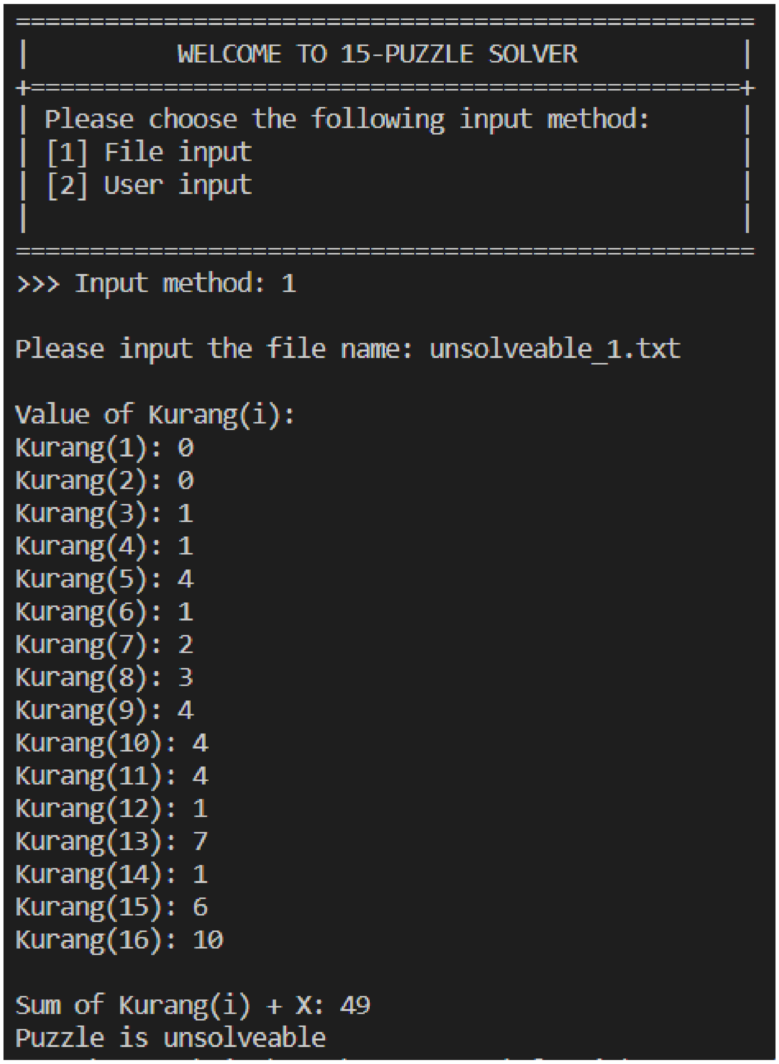


```
Total moves: 7

Initial Puzzle:

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

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

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

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



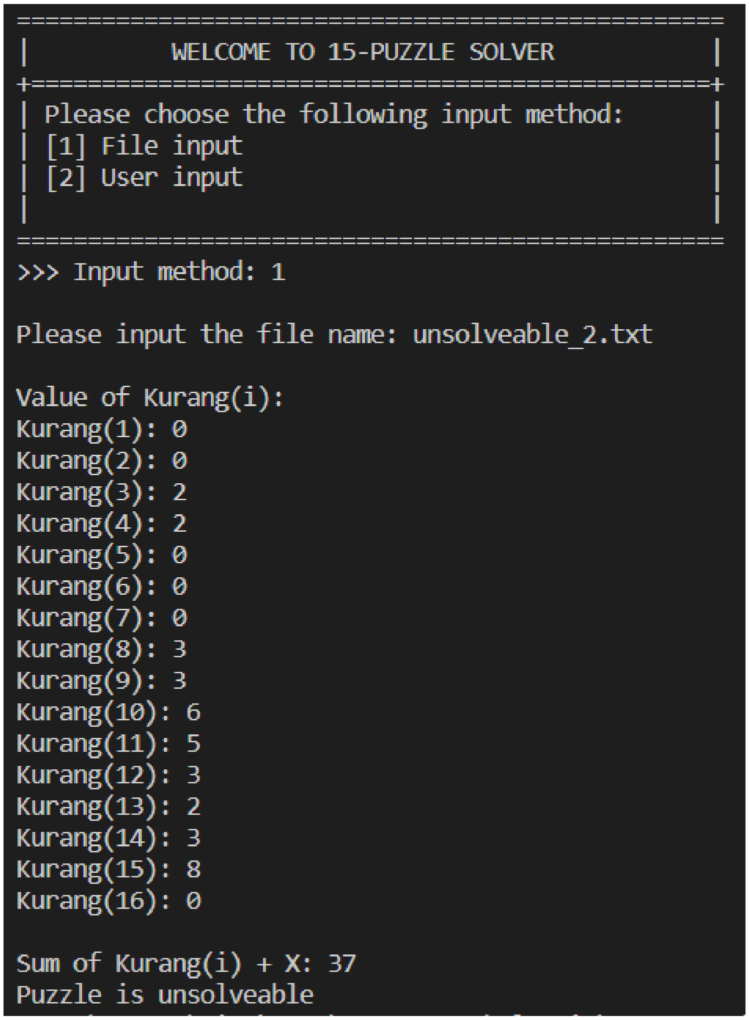
```
STEP 4: RIGHT
  1  2  3  4
  5  6  7  8
  9 10 15 11
 13 14  0 12

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

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

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

Number of generated nodes: 20
Execution time: 0.0016148090362548828 seconds
```

4. unsolveable_1.txt



```
==================================================
|          WELCOME TO 15-PUZZLE SOLVER           |
+================================================+
| Please choose the following input method:      |
| [1] File input                                 |
| [2] User input                                 |
|                                                |
==================================================
>>> Input method: 1

Please input the file name: unsolveable_1.txt

Value of Kurang(i):
Kurang(1): 0
Kurang(2): 0
Kurang(3): 1
Kurang(4): 1
Kurang(5): 4
Kurang(6): 1
Kurang(7): 2
Kurang(8): 3
Kurang(9): 4
Kurang(10): 4
Kurang(11): 4
Kurang(12): 1
Kurang(13): 7
Kurang(14): 1
Kurang(15): 6
Kurang(16): 10

Sum of Kurang(i) + X: 49
Puzzle is unsolveable
```

5. unsolveable_2.txt



```
==================================================
|          WELCOME TO 15-PUZZLE SOLVER           |
+================================================+
| Please choose the following input method:      |
| [1] File input                                 |
| [2] User input                                 |
|                                                |
==================================================
>>> Input method: 1

Please input the file name: unsolveable_2.txt

Value of Kurang(i):
Kurang(1): 0
Kurang(2): 0
Kurang(3): 2
Kurang(4): 2
Kurang(5): 0
Kurang(6): 0
Kurang(7): 0
Kurang(8): 3
Kurang(9): 3
Kurang(10): 6
Kurang(11): 5
Kurang(12): 3
Kurang(13): 2
Kurang(14): 3
Kurang(15): 8
Kurang(16): 0

Sum of Kurang(i) + X: 37
Puzzle is unsolveable
```

E. 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		✓

F. Alamat Drive Kode Program

https://github.com/Kyasaaa/Tucil-3-IF2211-Strategi-Algoritma