

LAPORAN TUGAS KECIL

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

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Rania Dwi Fadhilah (K1)

13520142



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

DAFTAR ISI

DAFTAR ISI.....	i
I. ALGORITMA PROGRAM.....	1
1.1 PENJELASAN ALGORITMA	1
2.1 IMPLEMENTASI	2
II. HASIL PERCOBAAN.....	10
2.1 INTERAKSI I/O.....	10
2.2 UJI COBA TXT	12
2.3 UJI COBA MANUAL.....	26
LAMPIRAN	ii

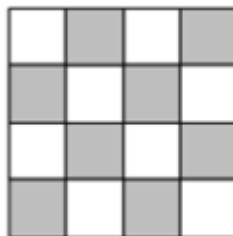
I. ALGORITMA PROGRAM

1.1 PENJELASAN ALGORITMA

Algoritma *Branch and Bound* merupakan algoritma yang kerap digunakan untuk menyelesaikan persoalan optimasi, dimana program harus meminimalkan atau memaksimalkan suatu fungsi objektif tertentu tanpa melanggar batasan persoalan. Pada dasarnya, algoritma ini merupakan gabungan dari algoritma BFS (*Breadth First Search*) dengan *least cost search*. Oleh karena itu, cara kerja algoritma ini adalah memberikan nilai *cost* pada setiap simpulnya, kemudian mengekskansi simpul berdasarkan urutan nilai *cost* yang paling kecil (apabila ingin meminimalkan) ataupun yang paling besar (apabila ingin memaksimalkan), bukan berdasarkan urutan pembangkitannya. Salah satu contoh aplikasi dari algoritma ini adalah pada permainan 15-puzzle. Pada tugas kecil ini, kita diminta untuk mengimplementasikan penyelesaian persoalan 15-Puzzle dengan menggunakan algoritma ini.

Berikut adalah cara kerja program *Branch and Bound* dalam menyelesaikan persoalan 15-puzzle :

1. Membaca *puzzle* menjadi *array* 4 x 4 dalam bentuk kelas *Puzzle* yang juga menyimpan nilai :
cost (harga jalur),
path (urutan jalur \rightarrow c/: [[UP], [LEFT]]),
letak basis,
bentuk atas, bawah, kanan, dan kirinya.
2. Menghitung nilai kurang(i) dari puzzle dengan cara menghitung jumlah elemen setelah posisi(i) yang memiliki nilai lebih dari elemen pada posisi(i).
3. Menghitung nilai X,



- a) Apabila basis berada di kotak terarsir, $X = 0$
 - b) Apabila basis berada di kotak tidak terarsir, $X = 1$
4. Apabila nilai kurang(i) + X = ganjil, maka *puzzle* tidak dapat diselesaikan
 5. Apabila nilai kurang(i) + X = genap, maka pencarian dimulai dengan menginisialisasi *heapqueue*, memasukkan puzzle acuan pertama ke *dictionary checked puzzle*.
 6. Apabila puzzle sudah dalam keadaan selesai, maka pencarian dihentikan dan lanjut ke tahap 9
 7. Apabila puzzle belum dalam keadaan selesai, maka akan dilakukan :
 - a) Inisialisasi *path* dengan path terakhir yang dilalui oleh puzzle untuk menghindari perpindahan balik, seperti sebelumnya UP kemudian DOWN dan sebelumnya LEFT kemudian RIGHT yang akan mengembalikan puzzle ke posisi semula
 - b) Apabila posisi baris basis tidak = 0 dan path terakhirnya bukan "DOWN", dilakukan pencarian ke atas dengan cara menginisialisasi *puzzle* baru dan menduplikasi matriks *puzzle* yang sedang dicari dan menukar posisi basisnya dengan matriks di atasnya.
 - c) Matriks diubah ke dalam bentuk list sementara untuk dicek apakah matriks tersebut sudah pernah diselidiki atau belum pada *dictionary checked puzzle*.
 - d) Apabila matriks puzzle belum pernah diselidiki, maka *path*-nya ditambahkan, basisnya diinisialisasi, dan *cost*-nya dihitung dengan rumus berikut :

$$c(P) = f(p) + g(p)$$

$f(p) = \text{panjang list (path)} = \text{panjang lintasan dari simpul akar ke } p$
 $g(p) = \text{jumlah elemen yang tidak berada}$
 $\text{pada posisinya (mengacu pada goal state)}$

- e) Puzzle dimasukkan pada *heapqueue* dengan *tuple value* berupa $\langle \text{cost Puzzle, Puzzle} \rangle$.
 - f) Apabila *puzzle* sudah selesai, maka pencarian dihentikan
 - g) Apabila posisi baris basis tidak = 3 dan path terakhirnya bukan “UP”, dilakukan pencarian ke bawah dengan cara menginisialisasi *puzzle* baru dan menduplikasi matriks *puzzle* yang sedang dicari dan menukar posisi basisnya dengan matriks di bawahnya. Kemudian, melakukan tahap c,d,e,f.
 - h) Apabila posisi kolom basis tidak = 0 dan path terakhirnya bukan “RIGHT”, dilakukan pencarian ke kiri dengan cara menginisialisasi *puzzle* baru dan menduplikasi matriks *puzzle* yang sedang dicari dan menukar posisi basisnya dengan matriks di kirinya. Kemudian, melakukan tahap c,d,e,f.
 - i) Apabila posisi kolom basis tidak = 3 dan path terakhirnya bukan “LEFT”, dilakukan pencarian ke kanan dengan cara menginisialisasi *puzzle* baru dan menduplikasi matriks *puzzle* yang sedang dicari dan menukar posisi basisnya dengan matriks di kanannya. Kemudian, melakukan tahap c,d,e,f.
 - j) Puzzle yang berada di *heapqueue* diangkat menjadi *puzzle* acuan. Puzzle yang diangkat merupakan *puzzle* dengan nilai *cost* terkecil yang ada pada *heapqueue*.
 - k) Puzzle acuan dijadikan list dan dimasukan ke *dictionary* checked puzzle.
8. Kembali ke langkah 6
9. Menuliskan solusi pada layar dengan cara mengiterasi *path* dari *puzzle* tersebut

2.1 IMPLEMENTASI

a. puzzle.py

```
import heapq
import numpy as np
import time

# PUZZLE CLASS
class Puzzle :
    goalstate = [[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]]

    # 1. Puzzle Initialization
    def __init__(self):
        self.mtx = np.empty((4,4), int) # Create Empty Matrix
        self.cost = 0 # Cost of the puzzle
        self.zero = (0,0) # Position of the basis
        self.path = [] # Path of the puzzle
        self.up, self.down, self.left, self.right = None, None, None, None # Up, Down,
Left, Right

    # 2. Cost Counter
    def countCost(self):
        count = 0
        for i in range (4) :
            for j in range (4) :
                # Count number of matrix that is not in position
                if (self.mtx[i][j] != Puzzle.goalstate[i][j]) :
```

```
        count += 1

    return count

# 3. Handle <
def __lt__(self, other) :
    return True

# IS FINISH BOOLEAN
def isFinish(puzzle):
    if (puzzle.countCost() == 0) :
        return True
    else :
        return False

# CHECK PUZZLE VALIDITY
def puzzleValid(puzzle) :
    # 1. Change 2d array -> list
    puzz = (np.reshape(puzzle.mtx, 16)).tolist()
    # 2. Check if 1-16 is in order
    for i in range (1,17) :
        if (SearchI(puzz, i) == -1) :
            return False

    return True

# READ PUZZLE FROM FILE
def readPuzzleFromFile(filename):
    # 1. Initialize Puzzle
    Puz15 = Puzzle()
    # 2. Open Puzzle File
    try :
        f = open(filename, "r")
        # 3. Read Puzzle
        try :
            for i in range(4):
                temp = f.readline()
                Puz15.mtx[i] = temp.split()
                for j in range(4):
                    Puz15.mtx[i][j] = int(Puz15.mtx[i][j])
                    if (Puz15.mtx[i][j] == 0 or Puz15.mtx[i][j] == 16) :
                        Puz15.mtx[i][j] = 16
                        Puz15.zero = (i,j)
            f.close()
            Puz15.cost = Puz15.countCost()

            # 4. Return Puzzle If Puzzle is Valid
            if (puzzleValid(Puz15)) :
                return Puz15
            else :
                print(" || Invalid ||")

input

        return None
    except :
```

```

        print(" || Invalid ||")
input
        return None
    except :
        print(" || File not found ||")
        return None

# READ PUZZLE FROM CONSOLE
def readPuzzleFromConsole():
    try :
        # 1. Initialize Puzzle
        Puz15 = Puzzle()
        # 2. Read Puzzle
        print(" || Input Puzzle (4x4), consists of ||")
:
        print(" || a. Number 1 - ||")
15
        print(" || b. 0 or 16 as ||")
basis
        for i in range(4):
            print(" >> ", end="")
            temp = str(input())
            Puz15.mtx[i] = temp.split()
            for j in range(4):
                Puz15.mtx[i][j] = int(Puz15.mtx[i][j])
                if (Puz15.mtx[i][j] == 0 or Puz15.mtx[i][j] == 16) :
                    Puz15.mtx[i][j] = 16
                    Puz15.zero = (i,j)

            Puz15.cost = Puz15.countCost()

        # 3. Return Puzzle If Puzzle is Valid
        if (puzzleValid(Puz15)) :
            return Puz15
        else :
            print(" || Invalid ||")
input
            return None
    except :
        print(" || Invalid ||")
input
        return None

# SEARCH INDEX OF NUM
def SearchI(puzz, num) :
    for i in range(len(puzz)) :
        if (puzz[i] == num) :
            return i

    return -1

# PRINT PUZZLE
def printPuzzle(puzz):
    print("

```

```

for i in range(4):
    print("                                ", end = "")
    for j in range(4):
        print("||", end="")
        if (puzz[i][j] == 16) :
            print(" ", end = "")
            continue
        print("%02d " % puzz[i][j], end = "")
    print("||")
    if(i != 3):
        print("
        print("
        print("                                ")

# CONVERT PUZZLE TO LIST
def puzzleToList(Puzzle) :
    puzz = ""
    temp = (np.reshape(Puzzle, 16)).tolist()
    for i in range(16) :
        puzz += str(temp[i])

    return puzz

# FIND KURANG(I) + X (BASIS POSITION)
# TO CHECK IF PUZZLE IS REACHABLE
def Reachable(Puzzle) :
    # 1. Convert Puzzle to List
    puzz = (np.reshape(Puzzle.mtx, 16)).tolist()

    # 2. Find Kurang(i)
    count = 0
    print("
    print("
    print("
    print("                                ")
    for i in range (1,17) :
        temp = 0
        index = SearchI(puzz, i)
        for j in range (index, 16) :
            if (i > puzz[j]) :
                temp += 1
        count += temp
        print("                                ||", end = "")
        if (i < 10) :
            print(" %d" % i, end = "")
        else :
            print("%d" % i, end = "")
        print(" ||", end="")
        if (temp < 10) :
            print(" %d" % temp, end = "")
        else :
            print("%d" % temp, end = "")
        print(" ||")
    print("
    print("                                ")

# 3. Find basis position
if ((Puzzle.zero[0] + Puzzle.zero[1]) % 2 != 0) :

```

```
        count += 1

    return count

# SOLVE PUZZLE
def solvePuzzle(PuzzPar) :
    # 1. Initialize Time
    now = time.time()
    # 2. Initialize Puzzle
    currPuzz = PuzzPar
    # 3. Initialize Checked Dictionary
    checked = {}
    puzz = puzzleToList(currPuzz.mtx)
    checked[puzz] = True
    # 4. Initialize HeapQueue
    pq = []
    # 5. Initialize node
    simpul = 0
    # 6. Loop until Puzzle = Goal State
    while (not (isFinish(currPuzz))) :
        # Check last path
        if (len(currPuzz.path) != 0) :
            path = currPuzz.path[len(currPuzz.path)-1]
        else :
            path = None

        # Check Up
        if (currPuzz.zero[0] != 0 and path != "DOWN") :
            currPuzz.up = Puzzle()
            currPuzz.up.mtx = np.copy(currPuzz.mtx)
            currPuzz.up.mtx[currPuzz.zero[0]][currPuzz.zero[1]] =
currPuzz.up.mtx[currPuzz.zero[0]-1][currPuzz.zero[1]]
            currPuzz.up.mtx[currPuzz.zero[0]-1][currPuzz.zero[1]] = 16
            temp = puzzleToList(currPuzz.up.mtx)
            if (temp not in checked) :
                currPuzz.up.path = currPuzz.path + ["UP"]
                currPuzz.up.zero = [currPuzz.zero[0]-1, currPuzz.zero[1]]
                currPuzz.up.cost = currPuzz.up.countCost() + len(currPuzz.up.path)
                heapq.heappush(pq, (currPuzz.up.cost, currPuzz.up))
                simpul += 1
            if (isFinish(currPuzz.up)) :
                currPuzz = currPuzz.up
                break

        # Check Down
        if (currPuzz.zero[0] != 3 and path != "UP") :
            currPuzz.down = Puzzle()
            currPuzz.down.mtx = np.copy(currPuzz.mtx)
            currPuzz.down.mtx[currPuzz.zero[0]][currPuzz.zero[1]] =
currPuzz.down.mtx[currPuzz.zero[0]+1][currPuzz.zero[1]]
            currPuzz.down.mtx[currPuzz.zero[0]+1][currPuzz.zero[1]] = 16
            temp = puzzleToList(currPuzz.down.mtx)
            if (temp not in checked) :
                currPuzz.down.path = currPuzz.path + ["DOWN"]
                currPuzz.down.zero = [currPuzz.zero[0]+1, currPuzz.zero[1]]
```



```
currPuzz.down.cost = currPuzz.down.countCost() + len(currPuzz.down.path)
heapq.heappush(pq, (currPuzz.down.cost, currPuzz.down))
simpul += 1
if (isFinish(currPuzz.down)) :
    currPuzz = currPuzz.down
    break

# Check Left
if (currPuzz.zero[1] != 0 and path != "RIGHT") :
    currPuzz.left = Puzzle()
    currPuzz.left.mtx = np.copy(currPuzz.mtx)
    currPuzz.left.mtx[currPuzz.zero[0]][currPuzz.zero[1]] =
currPuzz.left.mtx[currPuzz.zero[0]][currPuzz.zero[1]-1]
    currPuzz.left.mtx[currPuzz.zero[0]][currPuzz.zero[1]-1] = 16
    temp = puzzleToList(currPuzz.left.mtx)
    if (temp not in checked) :
        currPuzz.left.path = currPuzz.path + ["LEFT"]
        currPuzz.left.zero = [currPuzz.zero[0], currPuzz.zero[1]-1]
        currPuzz.left.cost = currPuzz.left.countCost() + len(currPuzz.left.path)
        heapq.heappush(pq, (currPuzz.left.cost, currPuzz.left))
        simpul += 1
        if (isFinish(currPuzz.left)) :
            currPuzz = currPuzz.left
            break

# Check Right
if (currPuzz.zero[1] != 3 and path != "LEFT") :
    currPuzz.right = Puzzle()
    currPuzz.right.mtx = np.copy(currPuzz.mtx)
    currPuzz.right.mtx[currPuzz.zero[0]][currPuzz.zero[1]] =
currPuzz.right.mtx[currPuzz.zero[0]][currPuzz.zero[1]+1]
    currPuzz.right.mtx[currPuzz.zero[0]][currPuzz.zero[1]+1] = 16
    temp = puzzleToList(currPuzz.right.mtx)
    if (temp not in checked) :
        currPuzz.right.path = currPuzz.path + ["RIGHT"]
        currPuzz.right.zero = [currPuzz.zero[0], currPuzz.zero[1]+1]
        currPuzz.right.cost = currPuzz.right.countCost() +
len(currPuzz.right.path)
        heapq.heappush(pq, (currPuzz.right.cost, currPuzz.right))
        simpul += 1
        if (isFinish(currPuzz.right)) :
            currPuzz = currPuzz.right
            break

currPuzz = heapq.heappop(pq)[1]
puzz = puzzleToList(currPuzz.mtx)
checked[puzz] = True

t = time.time() - now
# 7. Print Solution Puzzle
start = PuzzPar.mtx
zero = PuzzPar.zero
for i in range (len(currPuzz.path)) :
    # Change Position
    if (currPuzz.path[i] == "UP") :
```

```

        start[zero[0]][zero[1]] = start[zero[0]-1][zero[1]]
        start[zero[0]-1][zero[1]] = 16
        zero = (zero[0]-1, zero[1])
    elif (currPuzz.path[i] == "DOWN") :
        start[zero[0]][zero[1]] = start[zero[0]+1][zero[1]]
        start[zero[0]+1][zero[1]] = 16
        zero = (zero[0]+1, zero[1])
    elif (currPuzz.path[i] == "LEFT") :
        start[zero[0]][zero[1]] = start[zero[0]][zero[1]-1]
        start[zero[0]][zero[1]-1] = 16
        zero = (zero[0], zero[1]-1)
    elif (currPuzz.path[i] == "RIGHT") :
        start[zero[0]][zero[1]] = start[zero[0]][zero[1]+1]
        start[zero[0]][zero[1]+1] = 16
        zero = (zero[0], zero[1]+1)

    # Print Puzzle
    print("                                STEP %d = %s" % (i+1,
currPuzz.path[i]))
    printPuzzle(start)

    return currPuzz, t, simpul

```

b. main.py

```

import puzzle

def Mainmenu(first) :
    if first == True :
        print("=====
=====")
        print(" ||      1. Input From EXTERNAL FILE (inside the test
dir) ||      ||")
        print(" ||      2. Input From
CONSOLE ||      ||")
        print(" ||      3.
EXIT ||      ||")
        print(" ||      =====
||")
        try :
            op = int(input("                Choose Menu : "))
            if (op == 1) :
                filename = input("                File Path : ")
                Puz15 = puzzle.readPuzzleFromFile(".\\test\\" + filename)
            elif (op == 2) :
                Puz15 = puzzle.readPuzzleFromConsole()
            elif (op == 3) :
                Puz15 = None
            print("=====
=====")
        except :
            return op, Puz15
    except :

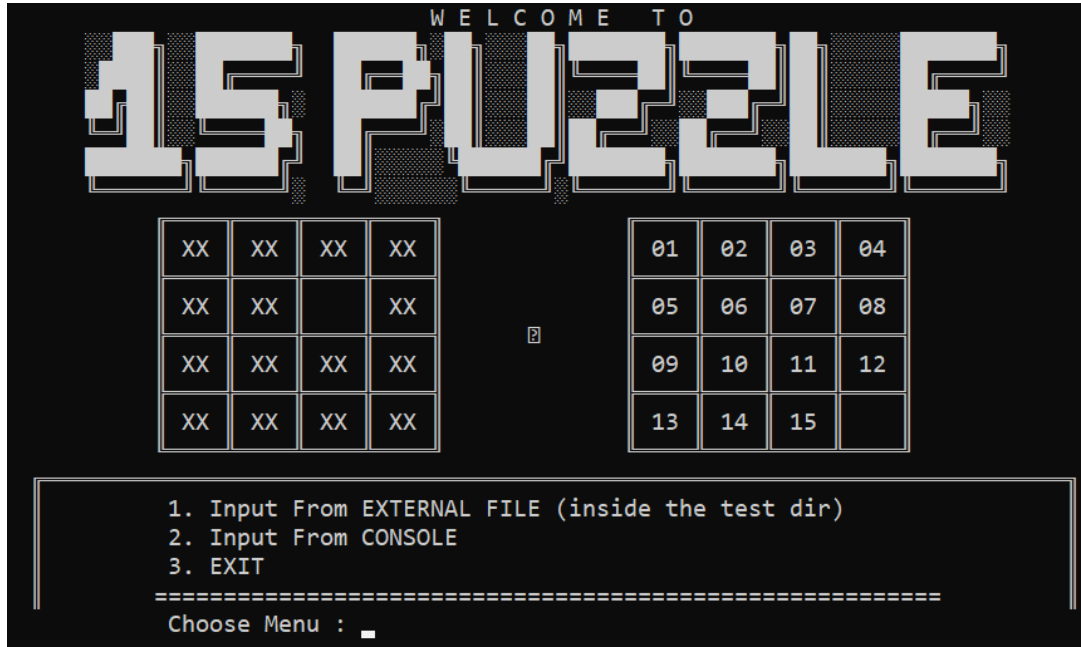
```

[illegible]

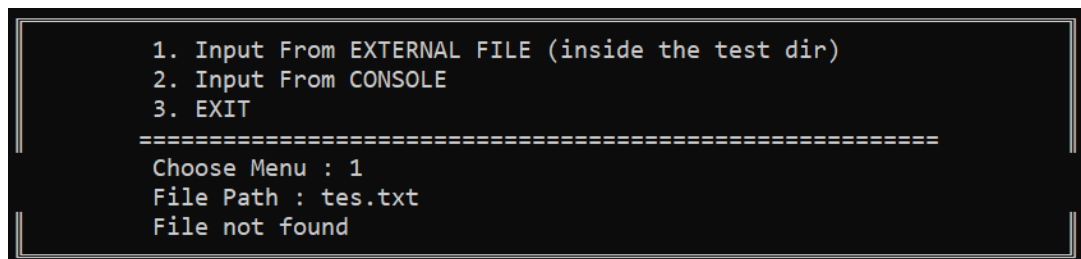
II. HASIL PERCOBAAN

2.1 INTERAKSI I/O

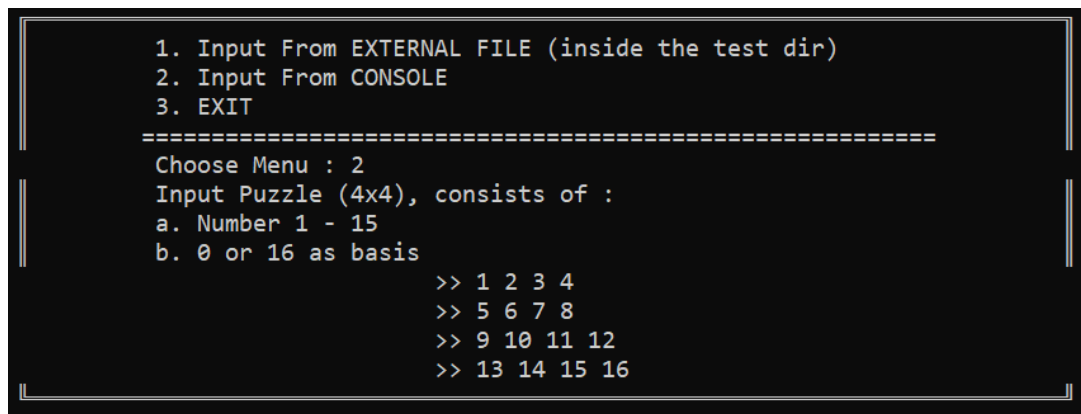
a. Interaksi *Input* Menu Awal



b. Interaksi *Input* Menu 1 (*External File*)



c. Interaksi *Input* Menu 2 (*Console*)



d. Interaksi *Output* Bagian Kurang(i)

P U Z Z L E			
01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

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

Kurang(i) = 0

e. Interaksi *Output* Bagian Solusi

===== S O L U T I O N =====			
STEP 1 = DOWN			
01	02	03	04
05	06	07	08
09	10		11
13	14	15	12

STEP 2 = RIGHT			
01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

STEP 3 = DOWN			
01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	
Node Count = 9			
Time = 0.000000 s			

2.2 UJI COBA TXT

Percobaan 1

Nama File	tc1easy.txt																																		
Bentuk Awal	1 2 3 4 5 6 16 8 9 10 7 11 13 14 15 12																																		
Hasil																																			
P U Z Z L E																																			
<table><tr><td>01</td><td>02</td><td>03</td><td>04</td></tr><tr><td>05</td><td>06</td><td></td><td>08</td></tr><tr><td>09</td><td>10</td><td>07</td><td>11</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>		01	02	03	04	05	06		08	09	10	07	11	13	14	15	12																		
01	02	03	04																																
05	06		08																																
09	10	07	11																																
13	14	15	12																																
<table><tr><td>i</td><td>kurang(i)</td></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>0</td></tr><tr><td>3</td><td>0</td></tr><tr><td>4</td><td>0</td></tr><tr><td>5</td><td>0</td></tr><tr><td>6</td><td>0</td></tr><tr><td>7</td><td>0</td></tr><tr><td>8</td><td>1</td></tr><tr><td>9</td><td>1</td></tr><tr><td>10</td><td>1</td></tr><tr><td>11</td><td>0</td></tr><tr><td>12</td><td>0</td></tr><tr><td>13</td><td>1</td></tr><tr><td>14</td><td>1</td></tr><tr><td>15</td><td>1</td></tr><tr><td>16</td><td>9</td></tr></table>		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
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																																		
Kurang(i) = 16																																			

<pre> ===== S O L U T I O N ===== STEP 1 = DOWN 01 02 03 04 05 06 07 08 09 10 11 13 14 15 12 STEP 2 = RIGHT 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 12 STEP 3 = DOWN 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 12 Node Count = 9 Time = 0.000000 s </pre>	
--	--

Percobaan 2

Nama File	tc2medium.txt
Bentuk Awal	<pre> 1 2 3 4 5 6 7 16 9 10 12 8 11 13 14 15 </pre>
Hasil	
<pre> P U Z Z L E 01 02 03 04 05 06 07 12 09 10 12 08 11 13 14 15 i kurang(i) 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 1 10 1 11 0 12 2 13 0 14 0 15 0 16 8 Kurang(i) = 12 </pre>	

===== S O L U T I O N =====
STEP 1 = DOWN

01	02	03	04
05	06	07	08
09	10	12	
11	13	14	15

STEP 2 = LEFT

01	02	03	04
05	06	07	08
09	10		12
11	13	14	15

STEP 3 = DOWN

01	02	03	04
05	06	07	08
09	10	14	12
11	13		15

STEP 4 = LEFT

01	02	03	04
05	06	07	08
09	10	14	12
11		13	15

STEP 5 = LEFT

01	02	03	04
05	06	07	08
09	10	14	12
	11	13	15

STEP 6 = UP

01	02	03	04
05	06	07	08
	10	14	12
09	11	13	15

STEP 7 = RIGHT

01	02	03	04
05	06	07	08
10		14	12
09	11	13	15

STEP 8 = DOWN

01	02	03	04
05	06	07	08
10	11	14	12
09		13	15

STEP 9 = RIGHT

01	02	03	04
05	06	07	08
10	11	14	12
09	13		15

STEP 10 = UP

01	02	03	04
05	06	07	08
10	11		12
09	13	14	15

STEP 11 = LEFT

01	02	03	04
05	06	07	08
10		11	12
09	13	14	15

STEP 12 = LEFT

01	02	03	04
05	06	07	08
	10	11	12
09	13	14	15

STEP 13 = DOWN

01	02	03	04
05	06	07	08
09	10	11	12
	13	14	15

STEP 14 = RIGHT

01	02	03	04
05	06	07	08
09	10	11	12
13		14	15

STEP 15 = RIGHT

01	02	03	04
05	06	07	08
09	10	11	12
13	14		15

STEP 16 = RIGHT

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Node Count = 1993

Time = 0.063008 s

Percobaan 3

Nama File	tc3intermediate.txt																																																		
Bentuk Awal	<div>2 3 4 11</div> <div>1 5 10 8</div> <div>9 6 12 15</div> <div>13 14 16 7</div>																																																		
Hasil																																																			
	<div>P U Z Z L E</div> <div><table><tr><td>02</td><td>03</td><td>04</td><td>11</td></tr><tr><td>01</td><td>05</td><td>10</td><td>08</td></tr><tr><td>09</td><td>06</td><td>12</td><td>15</td></tr><tr><td>13</td><td>14</td><td></td><td>07</td></tr></table></div> <div><table><tr><th>i</th><th>kurang(i)</th></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>1</td></tr><tr><td>4</td><td>1</td></tr><tr><td>5</td><td>0</td></tr><tr><td>6</td><td>0</td></tr><tr><td>7</td><td>0</td></tr><tr><td>8</td><td>2</td></tr><tr><td>9</td><td>2</td></tr><tr><td>10</td><td>4</td></tr><tr><td>11</td><td>7</td></tr><tr><td>12</td><td>1</td></tr><tr><td>13</td><td>1</td></tr><tr><td>14</td><td>1</td></tr><tr><td>15</td><td>3</td></tr><tr><td>16</td><td>1</td></tr></table></div> <div>Kurang(i) = 26</div>	02	03	04	11	01	05	10	08	09	06	12	15	13	14		07	i	kurang(i)	1	0	2	1	3	1	4	1	5	0	6	0	7	0	8	2	9	2	10	4	11	7	12	1	13	1	14	1	15	3	16	1
02	03	04	11																																																
01	05	10	08																																																
09	06	12	15																																																
13	14		07																																																
i	kurang(i)																																																		
1	0																																																		
2	1																																																		
3	1																																																		
4	1																																																		
5	0																																																		
6	0																																																		
7	0																																																		
8	2																																																		
9	2																																																		
10	4																																																		
11	7																																																		
12	1																																																		
13	1																																																		
14	1																																																		
15	3																																																		
16	1																																																		

===== S O L U T I O N =====
STEP 1 = RIGHT

02	03	04	11
01	05	10	08
09	06	12	15
13	14	07	

STEP 2 = UP

02	03	04	11
01	05	10	08
09	06	12	
13	14	07	15

STEP 3 = LEFT

02	03	04	11
01	05	10	08
09	06		12
13	14	07	15

STEP 4 = UP

02	03	04	11
01	05		08
09	06	10	12
13	14	07	15

STEP 5 = RIGHT

02	03	04	11
01	05	08	
09	06	10	12
13	14	07	15

STEP 6 = UP

02	03	04	
01	05	08	11
09	06	10	12
13	14	07	15

STEP 7 = LEFT

02	03		04
01	05	08	11
09	06	10	12
13	14	07	15

STEP 8 = LEFT

02		03	04
01	05	08	11
09	06	10	12
13	14	07	15

STEP 9 = LEFT

	02	03	04
01	05	08	11
09	06	10	12
13	14	07	15

STEP 10 = DOWN

01	02	03	04
	05	08	11
09	06	10	12
13	14	07	15

STEP 11 = RIGHT

01	02	03	04
05		08	11
09	06	10	12
13	14	07	15

STEP 12 = DOWN

01	02	03	04
05	06	08	11
09		10	12
13	14	07	15

STEP 13 = RIGHT

01	02	03	04
05	06	08	11
09	10		12
13	14	07	15

STEP 14 = DOWN

01	02	03	04
05	06	08	11
09	10	07	12
13	14		15

STEP 15 = RIGHT

01	02	03	04
05	06	08	11
09	10	07	12
13	14	15	

STEP 16 = UP

01	02	03	04
05	06	08	11
09	10	07	
13	14	15	12

STEP 18 = LEFT

01	02	03	04
05	06		08
09	10	07	11
13	14	15	12

STEP 19 = DOWN

01	02	03	04
05	06	07	08
09	10		11
13	14	15	12

STEP 20 = RIGHT

01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

STEP 21 = DOWN

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Node Count = 6283
Time = 0.215347 s

Percobaan 4

Nama File	tc4expert.txt
Bentuk Awal	1 2 3 4 5 6 11 15 9 14 13 10 16 7 8 12
Hasil	
<pre> P U Z Z L E 01 02 03 04 05 06 11 15 09 14 13 10 07 08 12 i kurang(i) 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 2 10 2 11 4 12 0 13 4 14 5 15 7 16 3 Kurang(i) = 28 ===== S O L U T I O N ===== STEP 1 = UP 01 02 03 04 05 06 11 15 14 13 10 09 07 08 12 STEP 2 = RIGHT 01 02 03 04 05 06 11 15 14 13 10 09 07 08 12 </pre>	

STEP 3 = RIGHT

01	02	03	04
05	06	11	15
14	13		10
09	07	08	12

STEP 4 = DOWN

01	02	03	04
05	06	11	15
14	13	08	10
09	07		12

STEP 5 = RIGHT

01	02	03	04
05	06	11	15
14	13	08	10
09	07	12	

STEP 6 = UP

01	02	03	04
05	06	11	15
14	13	08	
09	07	12	10

STEP 7 = UP

01	02	03	04
05	06	11	
14	13	08	15
09	07	12	10

STEP 8 = LEFT

01	02	03	04
05	06		11
14	13	08	15
09	07	12	10

STEP 9 = DOWN

01	02	03	04
05	06	08	11
14	13		15
09	07	12	10

STEP 10 = DOWN

01	02	03	04
05	06	08	11
14	13	12	15
09	07		10

STEP 11 = RIGHT

01	02	03	04
05	06	08	11
14	13	12	15
09	07	10	

STEP 12 = UP

01	02	03	04
05	06	08	11
14	13	12	
09	07	10	15

STEP 13 = LEFT

01	02	03	04
05	06	08	11
14	13		12
09	07	10	15

STEP 14 = DOWN

01	02	03	04
05	06	08	11
14	13	10	12
09	07		15

STEP 15 = LEFT

01	02	03	04
05	06	08	11
14	13	10	12
09		07	15

STEP 16 = UP

01	02	03	04
05	06	08	11
14		10	12
09	13	07	15

STEP 17 = LEFT

01	02	03	04
05	06	08	11
	14	10	12
09	13	07	15

STEP 18 = DOWN

01	02	03	04
05	06	08	11
09	14	10	12
	13	07	15

STEP 19 = RIGHT

01	02	03	04
05	06	08	11
09	14	10	12
13		07	15

STEP 20 = UP

01	02	03	04
05	06	08	11
09		10	12
13	14	07	15

STEP 21 = RIGHT

01	02	03	04
05	06	08	11
09	10		12
13	14	07	15

STEP 22 = DOWN

01	02	03	04
05	06	08	11
09	10	07	12
13	14		15

STEP 23 = RIGHT

01	02	03	04
05	06	08	11
09	10	07	12
13	14	15	

STEP 23 = RIGHT

01	02	03	04
05	06	08	11
09	10	07	12
13	14	15	

STEP 24 = UP

01	02	03	04
05	06	08	11
09	10	07	
13	14	15	12

STEP 25 = UP

01	02	03	04
05	06	08	
09	10	07	11
13	14	15	12

STEP 26 = LEFT

01	02	03	04
05	06		08
09	10	07	11
13	14	15	12

STEP 27 = DOWN

01	02	03	04
05	06	07	08
09	10		11
13	14	15	12

STEP 28 = RIGHT

01	02	03	04
05	06	07	08
09	10	11	
13	14	15	12

STEP 29 = DOWN

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	

Node Count = 1491324
Time = 82.026029 s

Percobaan 5

Nama File	tc5unsolved.txt
Bentuk Awal	<pre> 1 3 4 15 2 0 5 12 7 6 11 14 8 9 10 13 </pre>
Hasil	
<pre> P U Z Z L E 01 03 04 15 02 05 12 07 06 11 14 08 09 10 13 i kurang(i) 1 0 2 0 3 1 4 1 5 0 6 0 7 1 8 0 9 0 10 0 11 3 12 6 13 0 14 4 15 11 16 10 Kurang(i) = 37 Puzzle can not be solved !! </pre>	

Percobaan 6

Nama File	tc6error.txt
Bentuk Awal	<pre> 1 2 3 4 5 6 16 8 9 10 7 11 13 14 15 11 </pre>
Hasil	
<pre> 1. Input From EXTERNAL FILE (inside the test dir) 2. Input From CONSOLE 3. EXIT ===== Choose Menu : 1 File Path : tc6error.txt Invalid input </pre>	

2.3 UJI COBA MANUAL

Percobaan 7

Bentuk Awal		<div>>> 1 2 3 16 >> 5 6 7 4 >> 9 10 12 8 >> 13 14 11 15</div>																																			
Hasil																																					
	P U Z Z L E																																				
	<table><tr><td>01</td><td>02</td><td>03</td><td></td></tr><tr><td>05</td><td>06</td><td>07</td><td>04</td></tr><tr><td>09</td><td>10</td><td>12</td><td>08</td></tr><tr><td>13</td><td>14</td><td>11</td><td>15</td></tr></table>	01	02	03		05	06	07	04	09	10	12	08	13	14	11	15																				
01	02	03																																			
05	06	07	04																																		
09	10	12	08																																		
13	14	11	15																																		
	<table><tr><td>i</td><td>kurang(i)</td></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>0</td></tr><tr><td>3</td><td>0</td></tr><tr><td>4</td><td>0</td></tr><tr><td>5</td><td>1</td></tr><tr><td>6</td><td>1</td></tr><tr><td>7</td><td>1</td></tr><tr><td>8</td><td>0</td></tr><tr><td>9</td><td>1</td></tr><tr><td>10</td><td>1</td></tr><tr><td>11</td><td>0</td></tr><tr><td>12</td><td>2</td></tr><tr><td>13</td><td>1</td></tr><tr><td>14</td><td>1</td></tr><tr><td>15</td><td>0</td></tr><tr><td>16</td><td>12</td></tr></table>	i	kurang(i)	1	0	2	0	3	0	4	0	5	1	6	1	7	1	8	0	9	1	10	1	11	0	12	2	13	1	14	1	15	0	16	12		
i	kurang(i)																																				
1	0																																				
2	0																																				
3	0																																				
4	0																																				
5	1																																				
6	1																																				
7	1																																				
8	0																																				
9	1																																				
10	1																																				
11	0																																				
12	2																																				
13	1																																				
14	1																																				
15	0																																				
16	12																																				
	Kurang(i) = 22																																				
	===== S O L U T I O N =====																																				
	STEP 1 = DOWN																																				
	<table><tr><td>01</td><td>02</td><td>03</td><td>04</td></tr><tr><td>05</td><td>06</td><td>07</td><td></td></tr><tr><td>09</td><td>10</td><td>12</td><td>08</td></tr><tr><td>13</td><td>14</td><td>11</td><td>15</td></tr></table>	01	02	03	04	05	06	07		09	10	12	08	13	14	11	15																				
01	02	03	04																																		
05	06	07																																			
09	10	12	08																																		
13	14	11	15																																		
	STEP 2 = DOWN																																				
	<table><tr><td>01</td><td>02</td><td>03</td><td>04</td></tr><tr><td>05</td><td>06</td><td>07</td><td>08</td></tr><tr><td>09</td><td>10</td><td>12</td><td></td></tr><tr><td>13</td><td>14</td><td>11</td><td>15</td></tr></table>	01	02	03	04	05	06	07	08	09	10	12		13	14	11	15																				
01	02	03	04																																		
05	06	07	08																																		
09	10	12																																			
13	14	11	15																																		
	STEP 3 = LEFT																																				
	<table><tr><td>01</td><td>02</td><td>03</td><td>04</td></tr><tr><td>05</td><td>06</td><td>07</td><td>08</td></tr><tr><td>09</td><td>10</td><td></td><td>12</td></tr><tr><td>13</td><td>14</td><td>11</td><td>15</td></tr></table>	01	02	03	04	05	06	07	08	09	10		12	13	14	11	15																				
01	02	03	04																																		
05	06	07	08																																		
09	10		12																																		
13	14	11	15																																		

STEP 4 = DOWN			
01	02	03	04
05	06	07	08
09	10	11	12
13	14		15
STEP 5 = RIGHT			
01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	
Node Count = 11			
Time = 0.000968 s			

Percobaan 8

Bentuk Awal	<pre>>> 1 2 4 3 >> 7 8 0 5 >> 6 9 10 11 >> 12 13 14 15</pre>																																		
Hasil																																			
P U Z Z L E																																			
<table><tr><td>01</td><td>02</td><td>04</td><td>03</td></tr><tr><td>07</td><td>08</td><td></td><td>05</td></tr><tr><td>06</td><td>09</td><td>10</td><td>11</td></tr><tr><td>12</td><td>13</td><td>14</td><td>15</td></tr></table>		01	02	04	03	07	08		05	06	09	10	11	12	13	14	15																		
01	02	04	03																																
07	08		05																																
06	09	10	11																																
12	13	14	15																																
<table><tr><th>i</th><th>kurang(i)</th></tr><tr><td>1</td><td>0</td></tr><tr><td>2</td><td>0</td></tr><tr><td>3</td><td>0</td></tr><tr><td>4</td><td>1</td></tr><tr><td>5</td><td>0</td></tr><tr><td>6</td><td>0</td></tr><tr><td>7</td><td>2</td></tr><tr><td>8</td><td>2</td></tr><tr><td>9</td><td>0</td></tr><tr><td>10</td><td>0</td></tr><tr><td>11</td><td>0</td></tr><tr><td>12</td><td>0</td></tr><tr><td>13</td><td>0</td></tr><tr><td>14</td><td>0</td></tr><tr><td>15</td><td>0</td></tr><tr><td>16</td><td>9</td></tr></table>		i	kurang(i)	1	0	2	0	3	0	4	1	5	0	6	0	7	2	8	2	9	0	10	0	11	0	12	0	13	0	14	0	15	0	16	9
i	kurang(i)																																		
1	0																																		
2	0																																		
3	0																																		
4	1																																		
5	0																																		
6	0																																		
7	2																																		
8	2																																		
9	0																																		
10	0																																		
11	0																																		
12	0																																		
13	0																																		
14	0																																		
15	0																																		
16	9																																		
Kurang(i) = 15 Puzzle can not be solved !!																																			

LAMPIRAN

1. Repository Github :

<https://github.com/raniadf/15puzzle>

2. Berkas Teks :

tc1easy	1 2 3 4 5 6 16 8 9 10 7 11 13 14 15 12
tc2medium	1 2 3 4 5 6 7 16 9 10 12 8 11 13 14 15
tc3intermediate	2 3 4 11 1 5 10 8 9 6 12 15 13 14 16 7
tc4expert	1 2 3 4 5 6 11 15 9 14 13 10 16 7 8 12
tc5unsolved	1 3 4 15 2 0 5 12 7 6 11 14 8 9 10 13

3. Checklist :

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat menerima input dan menuliskan output	✓	
4. Luaran sudah benar untuk semua data uji	✓	
5. Bonus dibuat		✓