Laporan Tugas Kecil 3
IF2211 Strategi Algoritma

Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Disusun Oleh:
13520120 Afrizal Sebastian

Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
2021

DAFTAR ISI

BAB I
Algoritma *Branch and Bound*

1.1 Deskripsi Langkah-Langkah Penggunaan Algoritma *Branch and Bound* dalam penyelesaian persoalaan 15-Puzzle

Program memiliki 2 kelas utama, yaitu kelas Puzzle dan PriorityQueue. Kelas Puzzle digunakaan untuk menampung matriks 15-Puzzle tersebut, *depth*, *cost*, *lastMove*, *dan TotalMove*. lastMove digunakan untuk menyimpan gerakan yang digunakan untuk sampai ke state tersebut dan TotalMove digunakan untuk menyimpan semua gerakan yang digunakan untuk sampai ke state tersebut. Kelas PriorityQueue menampung sebuah array dan sebuah function. Array nantinya digunakan untuk meyimpan puzzle dan function digunakan untuk melakukan pengurutan saat memasukkan puzzle ke array berdasakan Prioritas.

Program akan menerima sebuah masukan berupa matriks 15-puzzle. Hasil dari masukan akan dimasukkan ke dalam array 2D. Array tersebut akan dilakukan pengecekan terlebih dahulu apakan puzzle dapat sampai ke state akhir atau tidak. Pengecekan digunakan dengan function isSolveAble() dengan melihat *Kurang(i) + X* adalah genap. Jika ganjil makan akan ditampilkan pesan, sedangkan genap akan melanjutkan proses penyelesaian puzzle.

Puzzle yang dapat diselesaikan akan melakukan inisialisai sebuah PriorityQueue dengan function *lambda x,y : x.cost <= y.cost* function tersebut digunakan untuk melakukan pengurutan saat memasukkan puzzle ke Queue. Array 2D puzzle diatas, akan dimasukkan kedalam sebuah kelas Puzzle dengan kedalaman 0. Pada kasus ini digunakan sebuah dictionary untuk dapat mentracking state-state mana saja yang telah di kunjungi akan tidak melakukan pengecekan ke state yang sama berulang kali.

Program akan melakukan looping hingga Queue kosong atau hingga goal state ditemukan. Jika goal state berada pada antrian pertama pada Queue makan pencarian akan di berhentikan. Jika tidak, makan antrian pertama pada Queue akan didequeue dari antrian. Puzzle tersebut akan dilakukakuan pergerakan untuk mendapatkan state selanjutnya. Jika state sudah ada pada dictionary maka tidak akan dienqueu ke dalam Queue, jika belum maka state yang baru akan dienqueue ke dalam Queue dan pada dictionary ditambahkan state baru. Hal tersebut dilakukan hingga menemukan goal state.

Jika goal state telah ditemukan, TotalMove pada goal state akan digunakan pada initialPuzzle untuk melakukan print ke console langkah-langkah penyelesaian yang diambil dari initial state hingga goal state.

BAB II
Source Code Program dengan Python

2.1 Puzzle

```
import copy
class Puzzle :

    def __init__(self, thePuzzle, depth):
        self.puzzle = thePuzzle
        self.depth = depth
        self.cost = 0
        self.lastMove = ""
        self.TotalMove = []

    def printPuzzle(self):
        print('-'*29)
        for i in range(4):
            for j in range(4):
                if(self.puzzle[i][j] >= 10):
                    if(j == 3):
                        if(self.puzzle[i][j] == 16):
                            print("|      |", end='\n')
                        else :
                            print("| " , self.puzzle[i][j]," |", end='\n')
                    else:
                        if(self.puzzle[i][j] == 16):
                            print("|      ", end='')
                        else :
                            print("| " , self.puzzle[i][j]," ", end='')
                else :
                    if(j == 3):
                        print("| ", self.puzzle[i][j], "  |", end='\n')
                    else:
                        print("| ", self.puzzle[i][j], "  ", end='')

            print('-'*29)

    def findEmptySlot(self):
        for i in range(4):
            for j in range(4):
                if(self.puzzle[i][j] == 16):
                    return (i,j)


    def isThereAreState(self, dict, matriks):
        for i in range(len(dict)):
            if(dict[i] == matriks):
                return True
        return False

    def move(self, row, column, move, dict):
        emptyRow, emptyCol = self.findEmptySlot()
        if(emptyRow+row>=0 and emptyRow+row<=3 and emptyCol+column>=0 and 
emptyCol+column<=3):
            newPuzzle = copy.deepcopy(self)
            newPuzzle.depth +=1
            newPuzzle.lastMove = move
            newPuzzle.TotalMove.append(move)
            newPuzzle.puzzle[emptyRow][emptyCol], 
newPuzzle.puzzle[emptyRow+row][emptyCol+column] = 
newPuzzle.puzzle[emptyRow+row][emptyCol+column], 
newPuzzle.puzzle[emptyRow][emptyCol]
            if(not self.isThereAreState(dict, newPuzzle.puzzle)):
                dict[len(dict)] = newPuzzle.puzzle
                return newPuzzle
            else:
                return None
        else:
            return None

    def funcG(self):
        finalState = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 
15, 16]]
        costG = 0

        for i in range(4):
            for j in range(4) :
                if(self.puzzle[i][j] != finalState[i][j] and 
self.puzzle[i][j] != 16):
                    costG += 1

        return costG

    def setCost(self):
        costG = self.funcG()
        self.cost = costG + self.depth

    def isGoalState(self):
        finalState = [[1,2,3,4], [5,6,7,8], [9,10,11,12], [13, 14, 15, 16]]
        return self.puzzle == finalState

    def printAll(self, totalMove):
        print("\nInitial State :")
        self.printPuzzle()

        moveUnit = [(-1,0), (0,-1), (1,0),(0,1)]

        for i in range(len(totalMove)):
            moveIdx = 0
            if(totalMove[i] == "Up"):
                moveIdx = 0
            elif(totalMove[i] == "Left"):
                moveIdx = 1
            elif(totalMove[i] == "Down"):
                moveIdx = 2
            elif(totalMove[i] == "Right"):
                moveIdx = 3

            row, col = moveUnit[moveIdx]
            emptyRow, emptyCol = self.findEmptySlot()
            self.puzzle[emptyRow][emptyCol], 
self.puzzle[emptyRow+row][emptyCol+col] = 
self.puzzle[emptyRow+row][emptyCol+col], self.puzzle[emptyRow][emptyCol]

            print("\nGerakan", i+1, ":",totalMove[i])
            self.printPuzzle()

        print("\nPuzzle Solved")
        print("The Step : ", end='')
        for i in range(len(totalMove)):
            print(totalMove[i], end=' ')
        print("\nStep count :", len(totalMove))
```

2.2 PriorityQueue

```
class PriorityQueue :

    def __init__(self, prioFunc) :
        self.buffer = []
        self.prioFunc = prioFunc

    def isEmpty(self):
        return len(self.buffer) == 0

    def first(self) :
        return self.buffer[0]

    def enqueue(self, puzzle):
        idx = 0
        found = False

        while(not found and idx < len(self.buffer)) :
            if(self.prioFunc(puzzle, self.buffer[idx])) :
                found = True
            else :
                idx +=1

        self.buffer.insert(idx, puzzle)

    def dequeue(self):
        self.buffer.pop(0)
```

2.3 Main

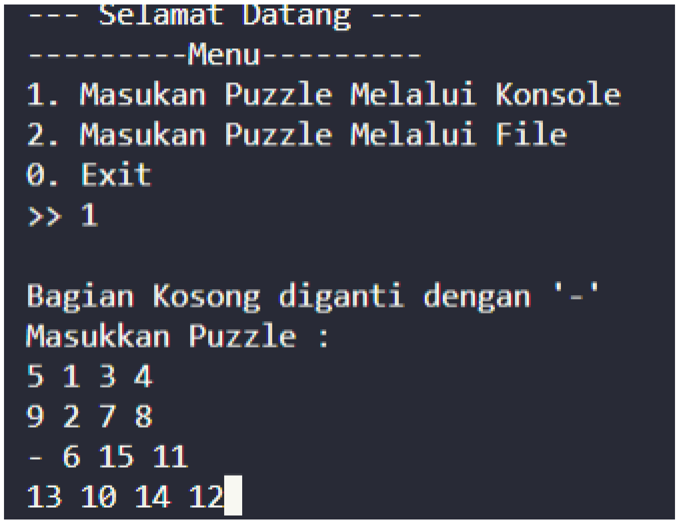
```
import timeit
from PriorityQueue import PriorityQueue
from Puzzle import Puzzle


def make1Dmatriks(matriks):
    idx = 0
    returnMat = [0 for i in range(16)]
    for i in range(4):
        for j in range(4) :
            returnMat[idx] = matriks[i][j]
            idx+=1
    return returnMat

def kurangI(matriks) :
    temp = make1Dmatriks(matriks)
    kurang = 0
    for i in range(len(temp)):
        for j in range(i+1, len(temp)):
            if(temp[i] > temp[j]):
                kurang +=1
    return kurang

def emptySlot(matriks):
    for i in range(4):
        for j in range(4):
            if(matriks[i][j] == 16):
                if(i+j) % 2 == 1:
                    return 1
                else:
                    return 0


def isSolveAble(matriks) :
    kurangi = kurangI(matriks)
    empty = emptySlot(matriks)
    return (kurangi + empty)%2 == 0


def isThereAreState(dict, matriks):
    for i in range(len(dict)):
        if(dict[i] == matriks):
            return True
    return False

def mainMenu():
    print("--- Selamat Datang ---")
    print("---------Menu---------")
    print("1. Masukan Puzzle Melalui Konsole")
    print("2. Masukan Puzzle Melalui File")
    print("0. Exit")
```

```
if __name__ == '__main__' :

    mainMenu()
    print(">>", end=" ")
    menu = int(input())
    while(menu != 1 and menu != 2 and menu !=0):
        print("Masukan Salah!\n")
        print(">>", end=" ")
        menu = int(input())



    print()
    initialState = [[0 for j in range(4)] for i in range(4)]
    if(menu == 1):
        print("Bagian Kosong diganti dengan '-' ")
        print("Masukkan Puzzle : ")
        for i in range(4):
            value = input("")
            temp = ""
            j = 0
            for k in range(len(value)):
                if(value[k] != ' '):
                    temp += value[k]
                else:
                    if(temp == '-'):
                        initialState[i][j] = 16
                        temp = ""
                        j+=1
                    else:
                        intTemp = int(temp)
                        initialState[i][j] = intTemp
                        temp = ""
                        j += 1

                if(k == len(value)-1):
                    if(temp == '-'):
                        initialState[i][j] = 16
                        temp = ""
                        j+=1
                    else:
                        intTemp = int(temp)
                        initialState[i][j] = intTemp
                        temp = ""
                        j += 1

    elif(menu == 2):
        path = "../test/"
        fileName = input("Masukkan nama file : ")
        path += fileName

        print()
        file = open(path, "r")
        line = file.readlines()
        for i in range(len(line)):
            value = line[i]
            temp = ""
            j = 0
            for k in range(len(value)):
                if(value[k] != ' '):
                    temp += value[k]
                else:
                    if(temp == '-'):
                        initialState[i][j] = 16
                        temp = ""
                        j+=1
                    else:
                        intTemp = int(temp)
                        initialState[i][j] = intTemp
                        temp = ""
                        j += 1

                if(k == len(value)-1):
                    if(temp == '-' or temp=='-\n'):
                        initialState[i][j] = 16
                        temp = ""
                        j+=1
                    else:
                        intTemp = int(temp)
                        initialState[i][j] = intTemp
                        temp = ""
                        j += 1
        file.close()
    elif(menu == 0):
        exit()

    print("Kurang(i) = ", kurangI(initialState)+emptySlot(initialState),"\n")

    startTime = timeit.default_timer()
    if(isSolveAble(initialState)):
        print("Puzzle is solveable")
        pQueue = PriorityQueue(lambda x,y : x.cost <= y.cost)

        moveUnit = [(-1,0), (0,-1), (1,0),(0,1)]
        moveName = ["Up", "Left", "Down", "Right"]
        moveOpposite = ["Down", "Right", "Up", "Left"]

        initialPuzzle = Puzzle(initialState, 0)
        pQueue.enqueue(initialPuzzle)

        stateTracking = {0 : initialPuzzle.puzzle}

        finished = False
        nodeCount = 0
        print("\nSolving.....")
        while(not pQueue.isEmpty() and not finished):
            if(pQueue.first().isGoalState()):
                finished = True
            else:
                current = pQueue.first()
                pQueue.dequeue()
                for i in range(len(moveName)):
                    if(current.lastMove == ""):
                        row, col = moveUnit[i]

                        nextPuzzle = current.move(row, col, moveName[i],
stateTracking)
                        if(nextPuzzle != None) :
                                nextPuzzle.setCost()
                                pQueue.enqueue(nextPuzzle)
                                nodeCount+=1

                    else:
                        lastMove = current.lastMove
                        idxMove = 0
                        found = False

                        while(not found and idxMove < len(moveName)):
                            if(moveName[idxMove] == lastMove):
                                found = True
                            else:
                                idxMove +=1

                        if(idxMove != lastMove):
                            row, col = moveUnit[i]

                            nextPuzzle = current.move(row, col, moveName[i],
stateTracking)
                            if(nextPuzzle != None) :
                                nextPuzzle.setCost()
                                pQueue.enqueue(nextPuzzle)
                                nodeCount+=1

        initialPuzzle.printAll(pQueue.first().TotalMove)
        print("\nRaised Node Count : ", nodeCount)
    else:
        print("Puzzle is unsolveable")
    stopTime = timeit.default_timer()

    timeExecustion = stopTime - startTime
    print("Execution Time :", timeExecustion, "seconds")
```

Bab III
Screeshot Hasil

3.1 Contoh Masukan

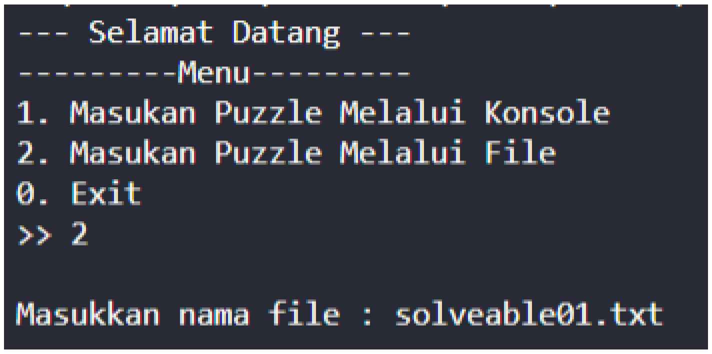
3.1.1 Masukan melalui console



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 1

Bagian Kosong diganti dengan '-'
Masukkan Puzzle :
5 1 3 4
9 2 7 8
- 6 15 11
13 10 14 12
```

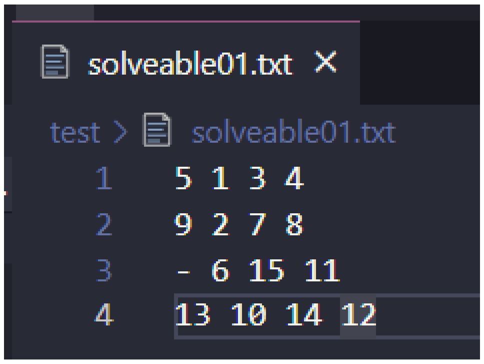
Gambar 3.1 : Masukkan Puzzle melalui Console

3.1.2 Masukkan melalui file



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : solveable01.txt
```

Gambar 3.2 : Masukkan Puzzle melalui file

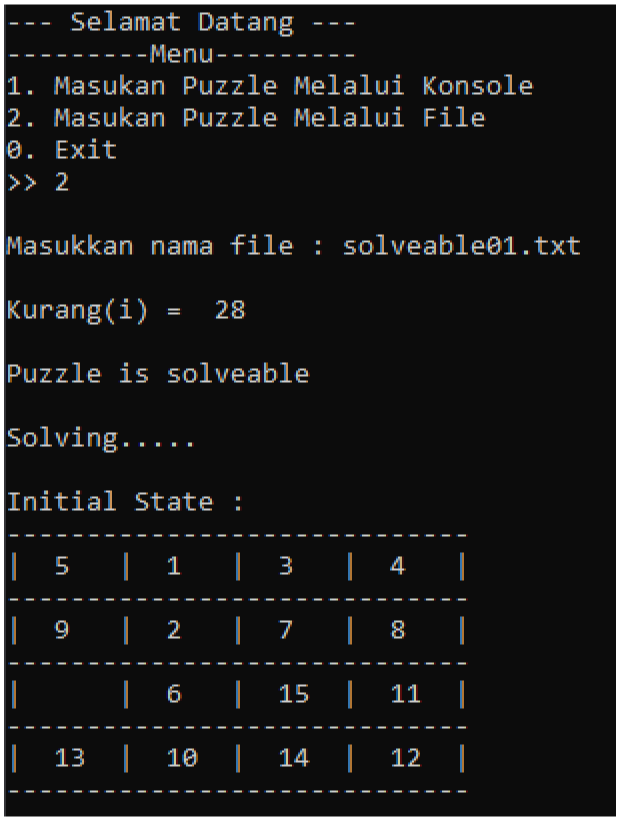


```
5 1 3 4
9 2 7 8
- 6 15 11
13 10 14 12
```

Gambar 3.3 : Contoh Puzzle pada file

3.2 TC-1 solveable01.txt

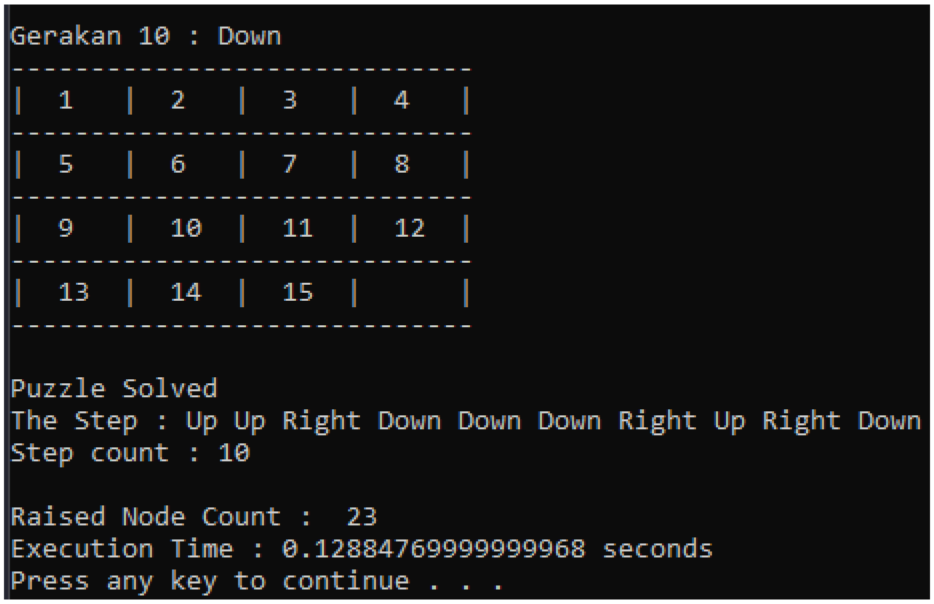
Awal



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : solveable01.txt

Kurang(i) =  28

Puzzle is solveable

Solving.....

Initial State :
-----------------------------
|  5   |  1   |  3   |  4   |
-----------------------------
|  9   |  2   |  7   |  8   |
-----------------------------
|      |  6   |  15  |  11  |
-----------------------------
|  13  |  10  |  14  |  12  |
-----------------------------
```

Gambar 3.4 : State Awal solveable01.txt

Akhir

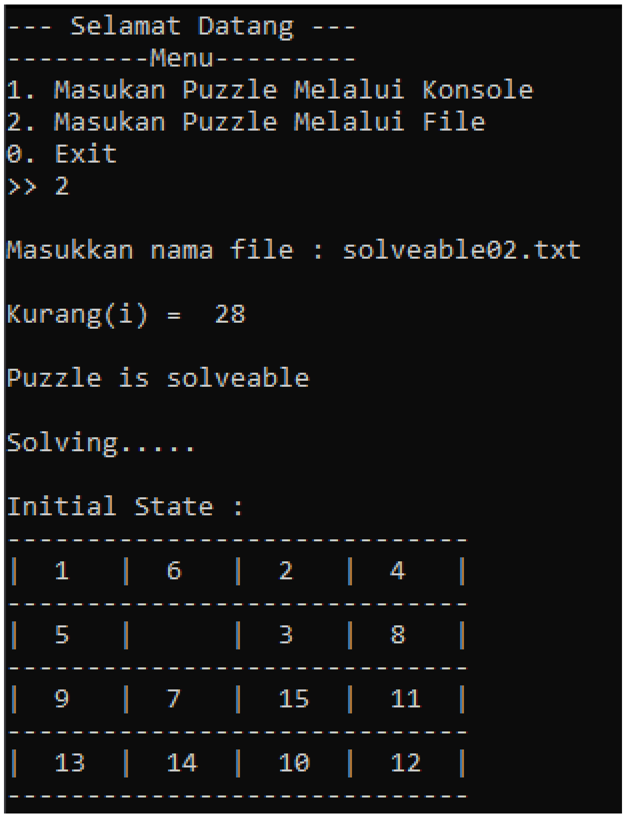


```
Gerakan 10 : Down
-----------------------------
|  1   |  2   |  3   |  4   |
-----------------------------
|  5   |  6   |  7   |  8   |
-----------------------------
|  9   |  10  |  11  |  12  |
-----------------------------
|  13  |  14  |  15  |      |
-----------------------------

Puzzle Solved
The Step : Up Up Right Down Down Down Right Up Right Down
Step count : 10

Raised Node Count :  23
Execution Time : 0.12884769999999968 seconds
Press any key to continue . . .
```

Gambar 3.5 : State Akhir solveable01.txt

3.3 TC-2 solveable02.txt

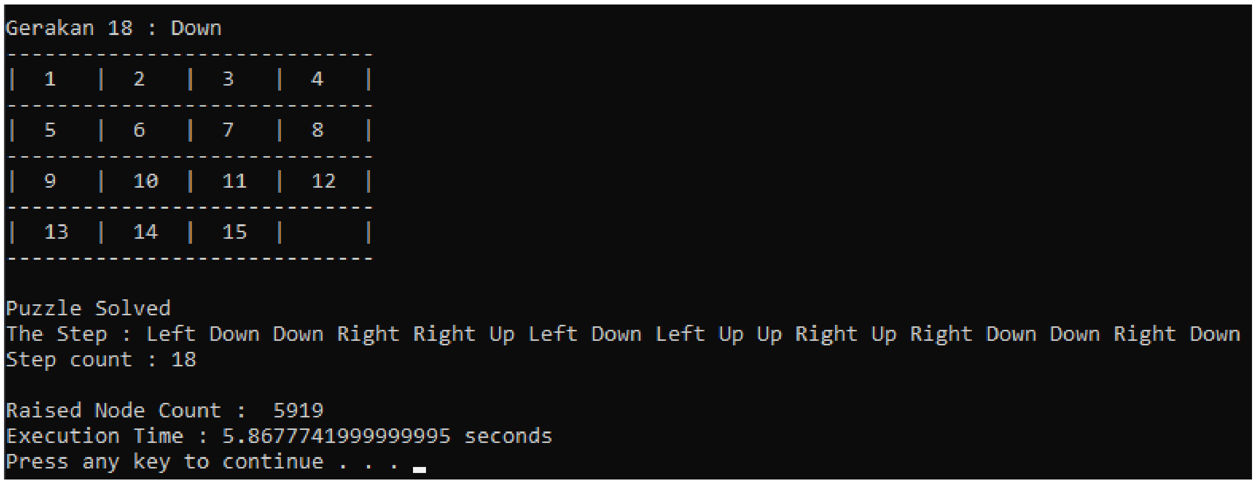
Awal



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : solveable02.txt

Kurang(i) =  28

Puzzle is solveable

Solving.....

Initial State :
-----------------------------
|  1   |  6   |  2   |  4   |
-----------------------------
|  5   |      |  3   |  8   |
-----------------------------
|  9   |  7   |  15  |  11  |
-----------------------------
|  13  |  14  |  10  |  12  |
-----------------------------
```

Gambar 3.6 : State Awal solveable02.txt

Akhir

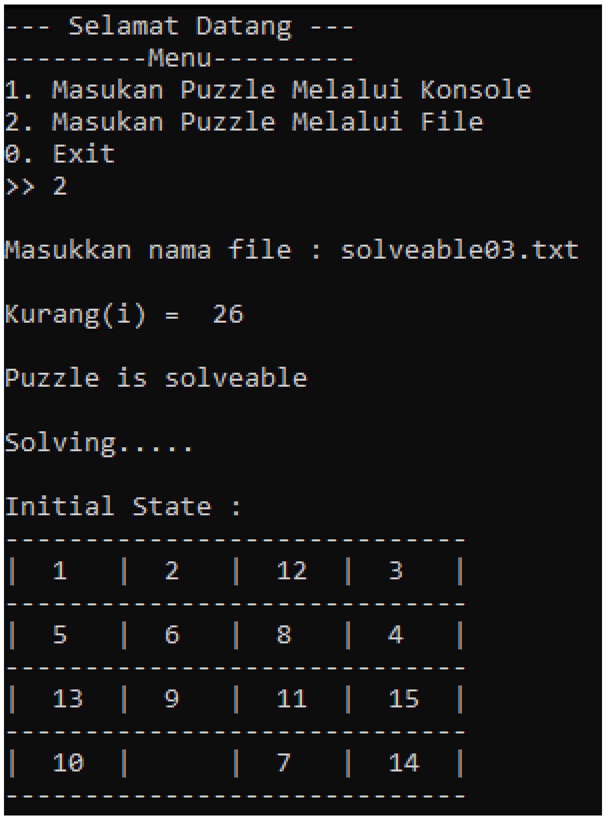


```
Gerakan 18 : Down
-----------------------------
|  1   |  2   |  3   |  4   |
-----------------------------
|  5   |  6   |  7   |  8   |
-----------------------------
|  9   |  10  |  11  |  12  |
-----------------------------
|  13  |  14  |  15  |      |
-----------------------------

Puzzle Solved
The Step : Left Down Down Right Right Up Left Down Left Up Up Right Up Right Down Down Right Down
Step count : 18

Raised Node Count :  5919
Execution Time : 5.8677741999999995 seconds
Press any key to continue . . .
```

Gambar 3.7 : State Akhir solveable02.txt

3.4 TC-3 solveable03.txt

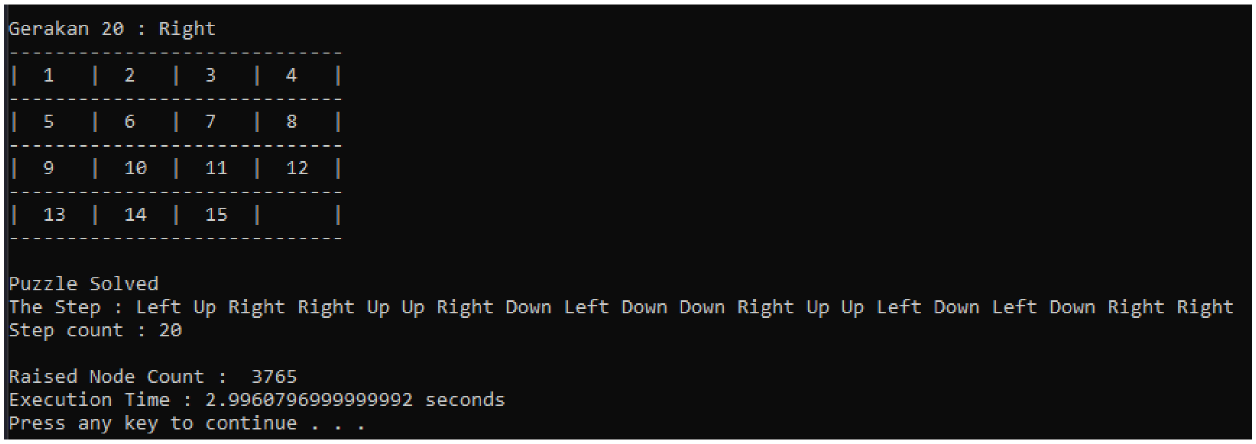
Awal



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : solveable03.txt

Kurang(i) =  26

Puzzle is solveable

Solving.....

Initial State :
-----------------------------
|  1   |  2   |  12  |  3   |
-----------------------------
|  5   |  6   |  8   |  4   |
-----------------------------
|  13  |  9   |  11  |  15  |
-----------------------------
|  10  |      |  7   |  14  |
-----------------------------
```

Gambar 3.8 : State Awal solveable03.txt

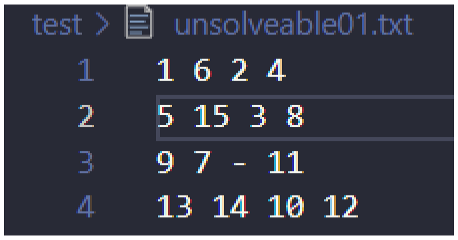
Akhir



```
Gerakan 20 : Right
-----------------------------
|  1   |  2   |  3   |  4   |
-----------------------------
|  5   |  6   |  7   |  8   |
-----------------------------
|  9   |  10  |  11  |  12  |
-----------------------------
|  13  |  14  |  15  |      |
-----------------------------

Puzzle Solved
The Step : Left Up Right Right Up Up Right Down Left Down Down Right Up Up Left Down Left Down Right Right
Step count : 20

Raised Node Count :  3765
Execution Time : 2.9960796999999992 seconds
Press any key to continue . . .
```

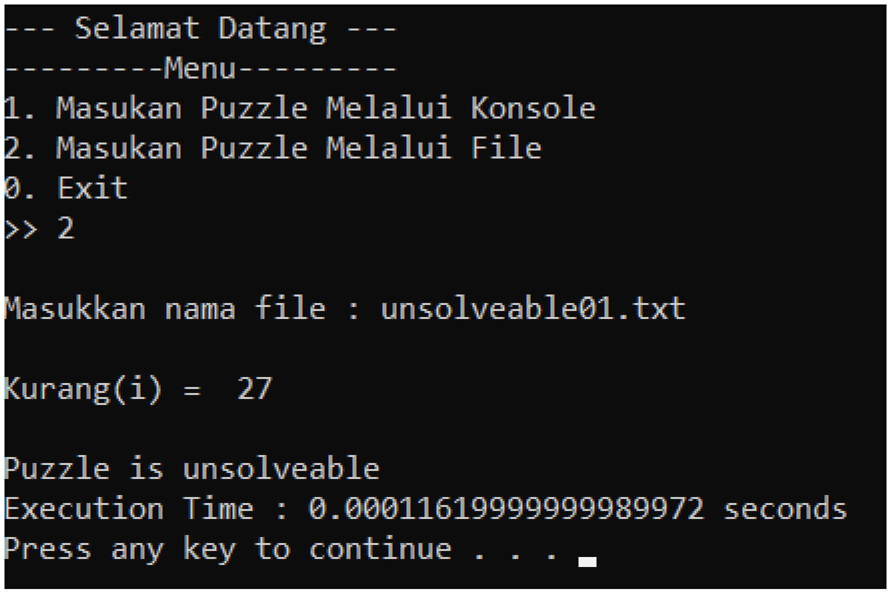
Gambar 3.9 : State Akhir solveable03.txt

3.5 TC-4 unsolveable01.txt



```
test > unsolveable01.txt
1 6 2 4
5 15 3 8
9 7 - 11
13 14 10 12
```

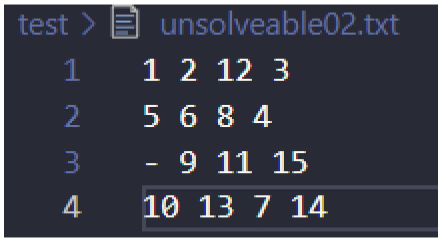
Gambar 3.10: Puzzle unsolveable01.txt



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : unsolveable01.txt

Kurang(i) =  27

Puzzle is unsolveable
Execution Time : 0.00011619999999989972 seconds
Press any key to continue . . .
```

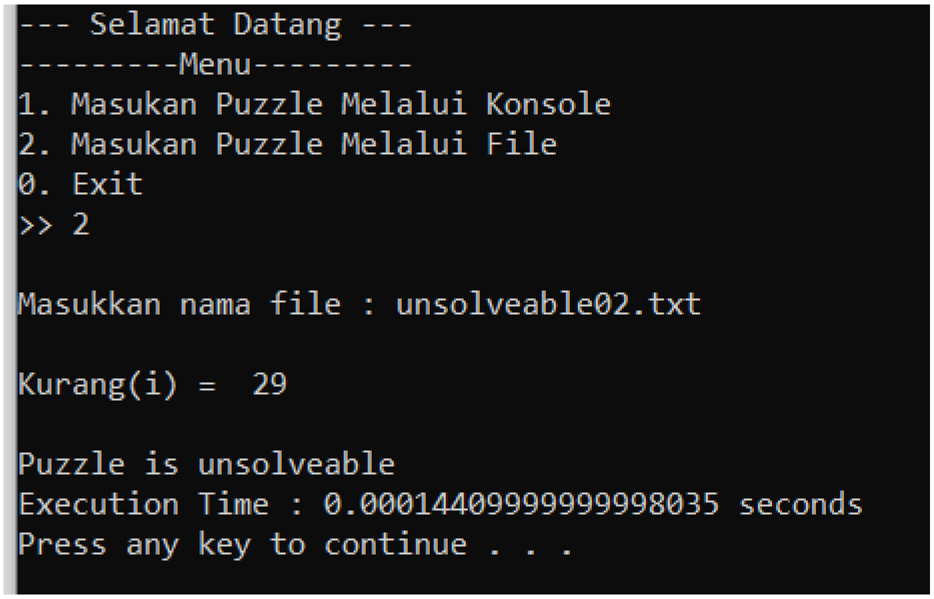
Gambar 3.11: Hasil unsolveable01.txt

3.6 TC-5 unsolveable02.txt



```
test > unsolveable02.txt
1 2 12 3
5 6 8 4
- 9 11 15
10 13 7 14
```

Gambar 3.11: Puzzle unsolveable02.txt



```
--- Selamat Datang ---
---------Menu---------
1. Masukan Puzzle Melalui Konsole
2. Masukan Puzzle Melalui File
0. Exit
>> 2

Masukkan nama file : unsolveable02.txt

Kurang(i) =  29

Puzzle is unsolveable
Execution Time : 0.00014409999999998035 seconds
Press any key to continue . . .
```

Gambar 3.13: Hasil unsolveable02.txt

3.7 Berkas Test Case

Tabel 3.1: Berkas Test Case

solveable01.txt	5 1 3 4 9 2 7 8 - 6 15 11 13 10 14 12
solveable02.txt	1 6 2 4 5 - 3 8 9 7 15 11 13 14 10 12
solveable03.txt	1 2 12 3 5 6 8 4 13 9 11 15 10 - 7 14
unsolveable01.txt	1 6 2 4 5 15 3 8 9 7 - 11 13 14 10 12
unsolveabl02.txt	1 2 12 3 5 6 8 4 - 9 11 15 10 13 7 14

Bab IV
Alamat GitHub

https://github.com/afrizalsebastian/Tucil3_13520120

CheckList

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		√