LAPORAN TUGAS KECIL III

“Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound”

Laporan Ini Dibuat untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Strategi Algoritma (IF2211)

KELAS 01

Dosen : Dr. Masayu Leylia Khodra, S.T., M.T.

Rio Alexander Audino / 13520088

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2021/2022

A. Cara Kerja Algoritima Branch and Bound

Program yang kali ini saya buat adalah implementasi pemecahan 15-Puzzle menggunakan algoritma Branch and Bound. Pencarian solusi 15-Puzzle dilakukan dengan algoritma Branch and Bound. Secara umum, terdapat beberapa langkah dalam implementasi algoritma ke dalam pemecahan masalah, yaitu:

a. Memvalidasi puzzle sebelum mulai diselesaikan
b. Jika puzzle dapat diselesaikan, puzzle dijadikan root node
c. Dimulai dari root node, menambahkan child node pada tiap gerakan ubin kosong memungkinkan.
d. Mengecek kondisi node apakah pernah ada/dibentuk sebelumnya
e. Melakukan kalkulasi *cost* pada tiap node yang terbentuk
f. Memilih node dengan dengan *cost* terkecil
g. Melakukan langkah c-e sampai menemukan node dengan ubin yang tersusun rapi
h. Ubin telah tersusun rapi dan 15-Puzzle berhasil dipecahkan

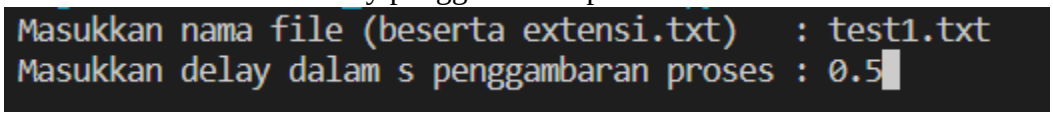
Secara detil, proses validasi dilakukan menggunakan fungsi KURANG. Fungsi KURANG akan memvalidasi puzzle melalui perbandingan posisi ubin sekarang dengan posisi ubin *goal*. Selain itu, proses perhitungan nilai *bound* atau *cost* didasarkan pada dua hal, yakni kedalaman node dan taksiran *cost* node sekarang sampai ke *goal*. Taksiran *cost* yang digunakan adalah jumlah ubin yang tak sesuai.

Selain proses di atas, terdapat juga beberapa hal yang membantu saya dalam implementasi algoritma BnB. Pada program ini, saya juga membuat dua kelas baru, Puzzle dan Node. Kelas Puzzle menangani segala proses yang terjadi pada ubin, seperti perhitungan taksiran cost, pergeseran ubin kosong, dll. Kelas Node menangani segala proses yang dibutuhkan oleh sebuah node, seperti penyimpanan informasi, proses output, serta pengecekan node.

B. Input-Output Program

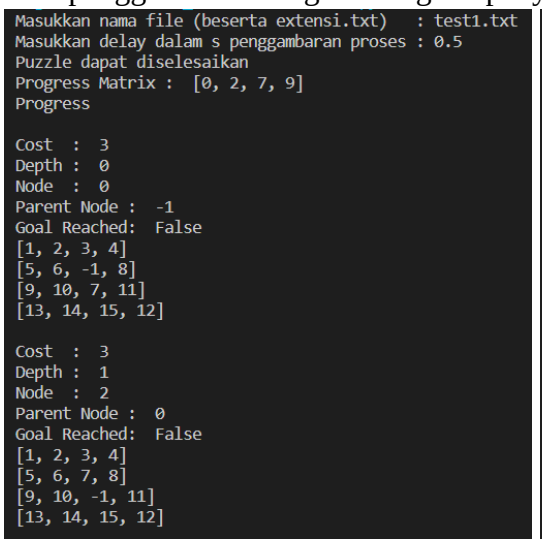
Contoh Input :

Memasukan nama file serta delay penggambaran proses di akhir

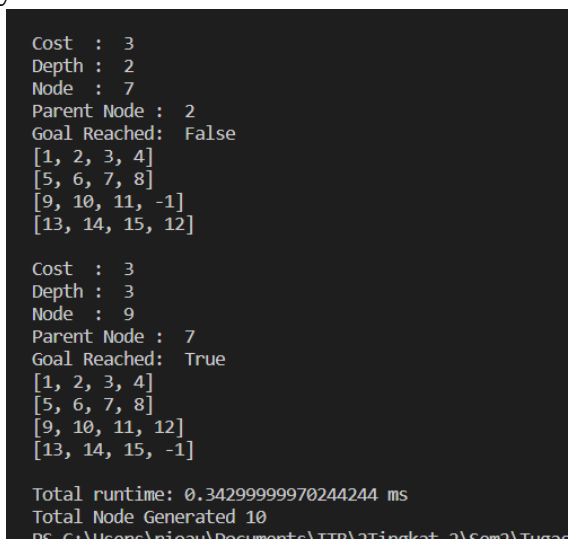


Contoh Output:

Proses penggambaran langkah-langkah penyelesaian dan hasil akhir



```
Masukkan nama file (beserta extensi.txt)   : test1.txt
Masukkan delay dalam s penggambaran proses : 0.5
Puzzle dapat diselesaikan
Progress Matrix :  [0, 2, 7, 9]
Progress

Cost  :  3
Depth :  0
Node  :  0
Parent Node :  -1
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, -1, 8]
[9, 10, 7, 11]
[13, 14, 15, 12]

Cost  :  3
Depth :  1
Node  :  2
Parent Node :  0
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, -1, 11]
[13, 14, 15, 12]
```



```
Cost  :  3
Depth :  2
Node  :  7
Parent Node :  2
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, -1]
[13, 14, 15, 12]

Cost  :  3
Depth :  3
Node  :  9
Parent Node :  7
Goal Reached:  True
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, -1]

Total runtime: 0.34299999970244244 ms
Total Node Generated 10
```

C. Checklist

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

D. Kode Program

File Main.py:

```python
import timeit

from Puzzle import *
from Node import *
from PuzzleCheck import *
from BnBAlgorithm import *
from PostProc import *

def readFromFile(fileName):
    puzzle = []
    fileName = "Test/" + fileName
    f = open(fileName, "r")
    Lines = f.readlines()

    for line in Lines:
        #puzzle.append(line.strip().split(" "))
        puzzle.append(list(map(int, line.strip().split(" "))))

    #print(puzzle)
    #print(type(puzzle[0][0]))

    return puzzle
```

```
def main():
    filename = input("Masukkan nama file (beserta extensi.txt)   : ")
    delay = float(input("Masukkan delay dalam s penggambaran proses : "))
    progress = []
    costRank =[]
    puzzle = Puzzle(readFromFile(filename))

    if (isReachableGoal(puzzle)):
        # Initiate values
        firstNode = Node(puzzle, 0, len(progress), -1)
        progress.append(firstNode)
        costRank = [0]

        # Run Algorithm and Check runtime
        start = timeit.default_timer()
        target = BnBAlgorithm(progress, costRank)
        stop = timeit.default_timer()

        printProgress(progress, findMatrixProgress(progress, target), stop-start, delay)

main()
```

file Puzzle.py:

```
class Puzzle:
    def __init__(self, models):
        self.models = [[-1 for _ in range(COLUMN)] for _ in range(ROW)]

        for i in range(ROW):
            for j in range(COLUMN):
                if (models[i][j] == -1):
                    emptyPos = [i, j]
                self.models[i][j] = models[i][j]

        self.startingEmptyPos = emptyPos
        self.emptyPos = emptyPos

    def getEmptyPos(self):
        return self.emptyPos

    def getModels(self):
        return self.models

    def moveEmptyPos(self, iEmpty, jEmpty, iTarget, jTarget):
        self.models[iEmpty][jEmpty] = self.models[iTarget][jTarget]
        self.models[iTarget][jTarget] = -1
        self.emptyPos = [iTarget, jTarget]
```

```python
def moveUp(self):
    i = self.emptyPos[0]
    j = self.emptyPos[1]

    if (i == 0):
        return False
    else:
        self.moveEmptyPos(i, j, i-1, j)
        return True

def moveLeft(self):
    i = self.emptyPos[0]
    j = self.emptyPos[1]

    if (j == 0):
        return False
    else:
        self.moveEmptyPos(i, j, i, j-1)
        return True

def moveRight(self):
    i = self.emptyPos[0]
    j = self.emptyPos[1]

    if (j == COLUMN-1):
        return False
    else:
        self.moveEmptyPos(i, j, i, j+1)
        return True

def moveDown(self):
    i = self.emptyPos[0]
    j = self.emptyPos[1]

    if (i == ROW-1):
        return False
    else:
        self.moveEmptyPos(i, j, i+1, j)
        return True
```

```python
def taksiranCostUntilGoal(self):
    totalCost = 0

    for i in range(ROW):
        for j in range(COLUMN):
            if (self.models[i][j] == -1):
                continue
            if (self.models[i][j] != (i*4 + j + 1)):
                totalCost += 1

    return totalCost

def isReachGoal(self):
    goalReaches = True
    count = 0

    for i in range(ROW):
        for j in range(COLUMN):
            count += 1

            if (self.models[i][j] != count and count < 16):
                goalReaches = False

    return goalReaches

def printInfo(self):
    print("Goal Reached: ", self.isReachGoal())
    for i in range(ROW):
        print(self.models[i])
```

File Node.py:

```python
class Node:
    def __init__(self, puzzle, depth, node, parentNode):
        self.puzzle = puzzle
        self.cost = puzzle.taksiranCostUntilGoal() + depth
        self.currCost = puzzle.taksiranCostUntilGoal()
        self.depth = depth
        self.node = node
        self.parentNode = parentNode

    def isNodeSame(self, other):
        isSame = True
        models = self.puzzle.getModels()
        otherModels = other.puzzle.getModels()
        for i in range(ROW):
            for j in range(COLUMN):
                if (models[i][j] != otherModels[i][j]):
                    isSame = False
                    break

        return isSame

    def getPuzzle(self):
        return self.puzzle

    def getCost(self):
        return self.cost
```

```python
def getCost(self):
    return self.cost

def getCurrCost(self):
    return self.currCost

def getDepth(self):
    return self.depth

def getNode(self):
    return self.node

def getParentNode(self):
    return self.parentNode

def printInfo(self):
    print("Cost  : ", self.cost)
    print("Depth : ", self.depth)
    print("Node  : ", self.node)
    print("Parent Node : ", self.parentNode)
    self.puzzle.printInfo()
    print()
```

File PuzzleCheck.py :

```python
def fungsiKurang(puzzle):
    # CONST
    greyArea = [1, 3, 4, 6, 9, 11, 12, 14]
    arrayLen = ROW*COLUMN

    # Convert matriks into array
    puzzleArray  = convertPuzzleToArray(puzzle)

    kurang = 0
    X = 0

    for i in range(arrayLen):
        kurangTemp = 0

        #Check empty position
        if (puzzleArray[i] == -1) :
            if (i in greyArea):
                X = 1
            kurangTemp = arrayLen - i - 1

        else:
            for j in range(i, arrayLen):
                if (puzzleArray[j] == -1):
                    continue

                elif (j > i and puzzleArray[j] < puzzleArray[i]):
                    kurangTemp += 1

        kurang += kurangTemp
        #print("Kurang", puzzleArray[i], "=", kurangTemp)

    # Add value of X
    kurang += X
    return kurang
```

```python
def convertPuzzleToArray(puzzle):
    models = puzzle.getModels()
    puzzleArray = []

    for i in range(ROW):
        for j in range(COLUMN):
            puzzleArray.append(models[i][j])

    return puzzleArray

def isReachableGoal(puzzle):
    value = fungsiKurang(puzzle)

    if (value % 2 == 0):
        print("Puzzle dapat diselesaikan")
        return True
    else:
        print("Puzzle tidak dapat diselesaikan")
        return False
```

File BnBAlgorithm.py:

```python
from Puzzle import *
from Node import *

def addNodes(progress, node, costRank):
    count = -1
    while(count < 4):
        count+=1

        addNode = False
        depth = node.getDepth()

        tempPuzzle = Puzzle(node.getPuzzle().getModels())
        if (count == 0 and tempPuzzle.moveUp()):
            addNode = True
        if (count == 2 and tempPuzzle.moveLeft()):
            addNode = True
        if (count == 3 and tempPuzzle.moveRight()):
            addNode = True
        if (count == 1 and tempPuzzle.moveDown()):
            addNode = True

        if addNode:
            newNode = Node(tempPuzzle, depth+1, len(progress), node.getNode())
            #Check if node already found before
            if (isNodeFoundBefor(progress, newNode)):
                continue

            progress.append(newNode)

            for i, iCost in enumerate(costRank):
                if (progress[iCost].getCost() > newNode.getCost()):
                    costRank.insert(i, len(progress)-1)
                    break

                if (i == len(costRank) - 1):
                    costRank.append(len(progress)-1)
                    break
```

```python
def isNodeFoundBefor(progress, otherNode):
    isFoundBefore = False
    for node in progress:
        if (node.getCurrCost() != otherNode.getCurrCost()):
            continue
        if (node.isNodeSame(otherNode)):
            isFoundBefore = True
            break

    return isFoundBefore

def BnBAlgorithm(progress, costRank):
    iNode = 0

    while(iNode < len(progress)):
        node = progress[costRank[0]]
        # Check if node is already target
        if (node.getPuzzle().isReachGoal()):
            return node

        addNodes(progress, node, costRank)
        costRank.pop(0)

        iNode+=1
```

File PostProc.py :

```python
import time
# Post Proccessing after solving the puzzle

def findMatrixProgress(progress, target):

    progressMatrix = [-1]
    tempNode = target

    while(True):
        currNode = tempNode.getNode()
        parentNode = tempNode.getParentNode()

        progressMatrix.insert(0, currNode)

        # Get Parent Node
        if (parentNode == -1):
            break
        else:
            tempNode = progress[parentNode]

    # Delete initial values
    progressMatrix.pop()
    print("Progress Matrix : ", progressMatrix)

    return progressMatrix

def printProgress(progress, progressMatrix, runTime, delay):
    print("Progress\n")

    for i in progressMatrix:
        time.sleep(delay)
        progress[i].printInfo()

    print("Total runtime:", runTime*1000, "ms")
    print("Total Node Generated", len(progress))
```

E. Testing

test1.txt :

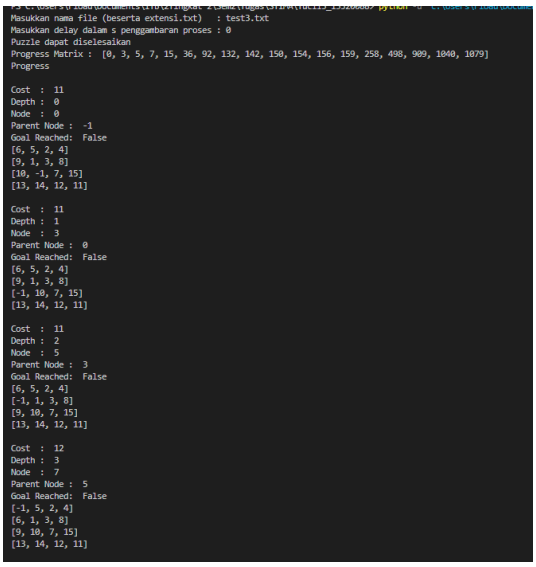
```
Masukkan nama file (beserta extensi.txt)   : test1.txt
Masukkan delay dalam s penggambaran proses : 0
Puzzle dapat diselesaikan
Progress Matrix :  [0, 2, 7, 9]
Progress

Cost  :  3
Depth :  0
Node  :  0
Parent Node :  -1
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, -1, 8]
[9, 10, 7, 11]
[13, 14, 15, 12]

Cost  :  3
Depth :  1
Node  :  2
Parent Node :  0
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, -1, 11]
[13, 14, 15, 12]

Cost  :  3
Depth :  2
Node  :  7
Parent Node :  2
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, -1]
[13, 14, 15, 12]

Cost  :  3
Depth :  3
Node  :  9
Parent Node :  7
Goal Reached:  True
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, -1]

Total runtime: 0.37960000190651044 ms
Total Node Generated 10
```

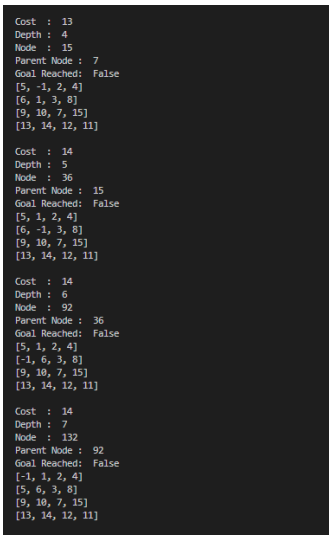
test2.txt :

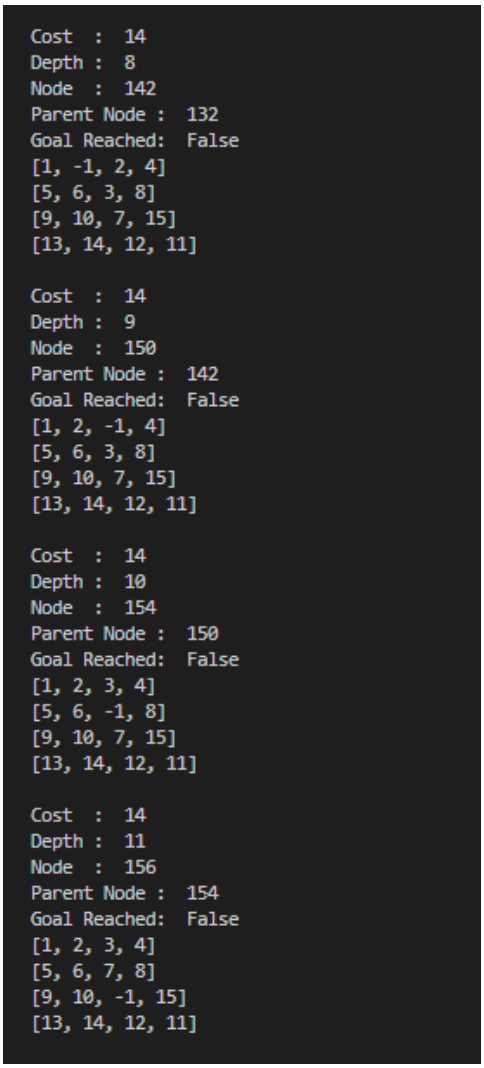
```
Masukkan nama file (beserta extensi.txt)   : test2.txt
Masukkan delay dalam s penggambaran proses : 0
Puzzle dapat diselesaikan
Progress Matrix :  [0, 1, 3, 5, 7, 9, 11]
Progress

Cost  :  6
Depth :  0
Node  :  0
Parent Node :  -1
Goal Reached:  False
[-1, 2, 3, 4]
[1, 6, 7, 8]
[5, 10, 11, 12]
[9, 13, 14, 15]

Cost  :  6
Depth :  1
Node  :  1
Parent Node :  0
Goal Reached:  False
[1, 2, 3, 4]
[-1, 6, 7, 8]
[5, 10, 11, 12]
[9, 13, 14, 15]

Cost  :  6
Depth :  2
Node  :  3
Parent Node :  1
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[-1, 10, 11, 12]
[9, 13, 14, 15]
```

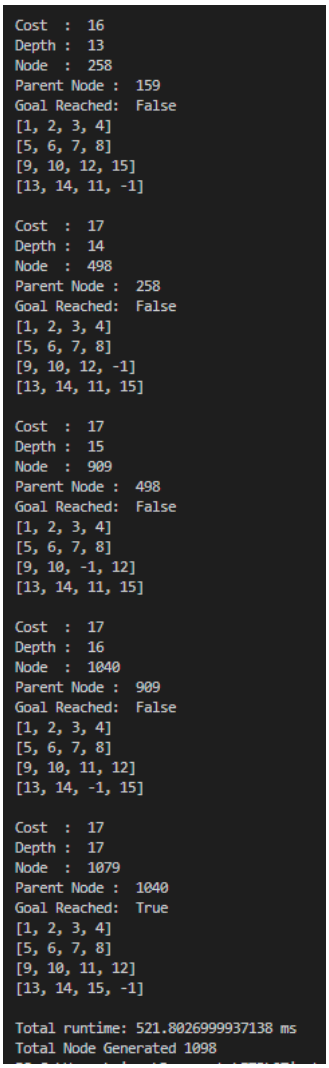
```
Cost  :  6
Depth :  3
Node  :  5
Parent Node :  3
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[-1, 13, 14, 15]

Cost  :  6
Depth :  4
Node  :  7
Parent Node :  5
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, -1, 14, 15]

Cost  :  6
Depth :  5
Node  :  9
Parent Node :  7
Goal Reached:  False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, -1, 15]

Cost  :  6
Depth :  6
Node  :  11
Parent Node :  9
Goal Reached:  True
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, -1]

Total runtime: 0.575300000491552 ms
Total Node Generated 12
```

test3.txt :


Masukkan nama file (beserta extensi.txt) : test3.txt
Masukkan delay dalam s penggambaran proses : 0
Puzzle dapat diselesaikan
Progress Matrix : [0, 3, 5, 7, 15, 36, 92, 132, 142, 150, 154, 156, 159, 258, 498, 909, 1040, 1079]
Progress
Cost : 11
Depth : 0
Node : 0
Parent Node : -1
Goal Reached: False
[6, 5, 2, 4]
[9, 1, 3, 8]
[10, -1, 7, 15]
[13, 14, 12, 11]
Cost : 11
Depth : 1
Node : 3
Parent Node : 0
Goal Reached: False
[6, 5, 2, 4]
[9, 1, 3, 8]
[-1, 10, 7, 15]
[13, 14, 12, 11]
Cost : 11
Depth : 2
Node : 5
Parent Node : 3
Goal Reached: False
[6, 5, 2, 4]
[-1, 1, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 12
Depth : 3
Node : 7
Parent Node : 5
Goal Reached: False
[-1, 5, 2, 4]
[6, 1, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]


Cost : 13
Depth : 4
Node : 15
Parent Node : 7
Goal Reached: False
[5, -1, 2, 4]
[6, 1, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 5
Node : 36
Parent Node : 15
Goal Reached: False
[5, 1, 2, 4]
[6, -1, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 6
Node : 92
Parent Node : 36
Goal Reached: False
[5, 1, 2, 4]
[-1, 6, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 7
Node : 132
Parent Node : 92
Goal Reached: False
[-1, 1, 2, 4]
[5, 6, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]


Cost : 14
Depth : 8
Node : 142
Parent Node : 132
Goal Reached: False
[1, -1, 2, 4]
[5, 6, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 9
Node : 150
Parent Node : 142
Goal Reached: False
[1, 2, -1, 4]
[5, 6, 3, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 10
Node : 154
Parent Node : 150
Goal Reached: False
[1, 2, 3, 4]
[5, 6, -1, 8]
[9, 10, 7, 15]
[13, 14, 12, 11]
Cost : 14
Depth : 11
Node : 156
Parent Node : 154
Goal Reached: False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, -1, 15]
[13, 14, 12, 11]


Cost : 16
Depth : 13
Node : 258
Parent Node : 159
Goal Reached: False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 12, 15]
[13, 14, 11, -1]
Cost : 17
Depth : 14
Node : 498
Parent Node : 258
Goal Reached: False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 12, -1]
[13, 14, 11, 15]
Cost : 17
Depth : 15
Node : 909
Parent Node : 498
Goal Reached: False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, -1, 12]
[13, 14, 11, 15]
Cost : 17
Depth : 16
Node : 1040
Parent Node : 909
Goal Reached: False
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, -1, 15]
Cost : 17
Depth : 17
Node : 1079
Parent Node : 1040
Goal Reached: True
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, -1]
Total runtime: 521.8026999937138 ms
Total Node Generated 1098

test4.txt :

```
Masukkan nama file (beserta extensi.txt)   : test4.txt
Masukkan delay dalam s penggambaran proses : 0
Puzzle tidak dapat diselesaikan
```

test5.txt :

```
Masukkan nama file (beserta extensi.txt)   : test5.txt
Masukkan delay dalam s penggambaran proses : 0
Puzzle tidak dapat diselesaikan
```

F. Sumber Kode Program

Link Github : https://github.com/Audino723/Tucil3_13520088