Laporan

Tugas Kecil 3 IF2211 Strategi Algoritma

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

Disusun Oleh:

Suryanto 13520059

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

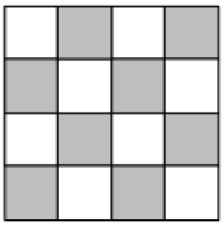
DAFTAR ISI

ALGORITMA BRANCH AND BOUND

Algoritma *Branch and Bound* adalah algoritma yang menggunakan pendekatan algoritma *Breadth First Search (*BFS) dan *least cost search*. Setiap simpul yang dibangkitkan akan memiliki nilai *cost* yang merupakan nilai taksiran lintasan termurah dari suatu simpul ke simpul tujuan. Pemeriksaan simpul pun tidak ditentukan berdasarkan urutan pembangkitan, melainkan nilai *cost* terkecil yang dimiliki oleh suatu simpul. Oleh karena itu, algoritma ini kerap digunakan untuk menyelesaikan persoalan optimasi yang meminimalkan atau memaksimalkan suatu fungsi objektif tanpa melanggar batasan persoalan. Salah satu dari implementasi dari algoritma ini adalah pencarian solusi pada permainan 15-Puzzle.

Penyelesaian persoalan 15-Puzzle pada program ini mengikuti tahapan-tahapan berikut ini :

1. Program membaca masukan dari file.txt pada folder test dan menyimpan nilai setiap bilangan ke dalam matriks berukuran 4 x 4. Alternatif : Puzzle di-*generate* secara *random* oleh program.
2. Selanjutnya akan dihitung nilai Kurang(*i*) dari puzzle, yaitu banyaknya bilangan $j<i$ yang memiliki posisi(*j*) > posisi(*i*).
3. Penghitungan nilai X dengan mengikuti kondisi di bawah ini,



   a) X = 1 apabila bilangan 16 (ubin kosong) terletak pada kotak yang terarsir.
   b) X = 0 pada kondisi sebaliknya.
4. Apabila nilai dari Σkurang(*i*) + X = ganjil, maka puzzle tidak dapat diselesaikan dan program akan menuliskan pesan puzzle tidak dapat diselesaikan. Namun, apabila bernilai genap, maka lanjutkan ke tahap ke-*5.*
5. Diinisiasikan *nodes* sebagai *list of list* yang berisi parentId, nodeId, matriks, cost, depth, dan move yang dilakukan.
6. Program akan membangkitkan simpul dengan melakukan pengecekan matriks ke-empat arah, yaitu atas, kiri, bawah, dan kanan. Namun apabila pergeseran ubin menghasilkan matriks yang sudah pernah ada sebelumnya, maka simpul tidak akan dimasukkan ke dalam antrian simpul yang dibangkitkan.
7. Melakukan pengecekan ke simpul dengan *cost* terendah yang merupakan total banyaknya ubin yang posisinya tidak sesuai dijumlahkan dengan banyaknya langkah yang harus diambil dari kondisi awal untuk menuju simpul (*depth)*
8. Tahapan 6 dan 7 akan terus dilakukan hingga matriks mencapai posisi yang ingin dituju, yaitu setiap ubin terurut membesar dari 1 hingga 15 (dengan posisi ubin kosong terletak di akhir).
9. Untuk penulisan *move* yang diperlukan, dilakukan pencarian simpul menggunakan *parentID* yang dimiliki oleh simpul akhir. Hal ini akan terus dilakukan hingga *parentID* bernilai 0 (simpul awal yang merepresentasikan kondisi awal puzzle)

SOURCE CODE

1. solver.py

```
import random

# Fungsi untuk mengubah matriks ke dalam string
def toString(mtrx):
    word = ''
    for x in mtrx:
        for y in x:
            word += "-" + (str(y))
    return word

# Fungsi untuk melakukan peng-copy-an matriks
def CopyMtrx(mtrx):
    temp = [[0 for i in range (4)] for j in range (4)]
    for i in range(4):
        for j in range(4):
            temp[i][j] = mtrx[i][j]
    return temp

# fungsi untuk membuat matriks secara random
def RandomizeMatrix():
    temp = random.sample(range(0,50), 50)
    mtrx = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
    for i in range (len(temp)):
        if temp[i] % 4 == 0:
            mtrx = Up(mtrx)
        elif temp[i] % 4 == 1:
            mtrx = Left(mtrx)
        elif temp[i] % 4 == 2:
            mtrx = Down(mtrx)
        else:
            mtrx = Right(mtrx)

    return mtrx

# Prosedur untuk mencetak matriks ke layar
def Output(mtrx):

    for i in range (4):
        for j in range(4):
            if (mtrx[i][j] != 16):
                print(str(mtrx[i][j]).rjust(4),end="")
            else:
                print(str("-").rjust(4),end="")
        print()
```

```
# Fungsi untuk mendapatkan posisi value pada matriks
def Position(mtrx, value):
    for i in range(4):
        for j in range(4):
            if value == mtrx[i][j]:
                return [i,j]

# Fungsi yang mengembalikan matriks yang tile #16 dimove ke atas
def Up(mtrx):
    nol = Position(mtrx, 16)
    tempMtrx = CopyMtrx(mtrx)

    if(nol[0] != 0 ):
        tempMtrx[nol[0]][nol[1]] = mtrx[nol[0]-1][nol[1]]
        tempMtrx[nol[0]-1][nol[1]] = 16

    return tempMtrx

# Fungsi yang mengembalikan matriks yang tile #16 dimove ke bawah
def Down(mtrx):
    nol = Position(mtrx, 16)
    tempMtrx = CopyMtrx(mtrx)

    if(nol[0] != 3 ):

        tempMtrx[nol[0]][nol[1]] = mtrx[nol[0]+1][nol[1]]
        tempMtrx[nol[0]+1][nol[1]] = 16

    return tempMtrx

# Fungsi yang mengembalikan matriks yang tile #16 dimove ke kiri
def Left(mtrx):
    nol = Position(mtrx, 16)

    tempMtrx = CopyMtrx(mtrx)

    if(nol[1] != 0 ):
        tempMtrx[nol[0]][nol[1]] = mtrx[nol[0]][nol[1]-1]
        tempMtrx[nol[0]][nol[1]-1] = 16

    return tempMtrx

# Fungsi yang mengembalikan matriks yang tile #16 dimove ke kanan
def Right(mtrx):
    nol = Position(mtrx, 16)
    tempMtrx = CopyMtrx(mtrx)
```

```
    if(nol[1] != 3 ):
        tempMtrx[nol[0]][nol[1]] = mtrx[nol[0]][nol[1]+1]
        tempMtrx[nol[0]][nol[1]+1] = 16

    return tempMtrx

# Fungsi untuk menghitung fungsi kurang pada tiap tile
def Kurang(mtrx, num):
    count = 0
    posisi = Position(mtrx, num)

    for j in range (posisi[1], 4):
        if (num > mtrx[posisi[0]][j]):
            count+=1

    for i in range (posisi[0]+1, 4):
        for j in range(4):

            if (num > mtrx[i][j]):
                count+=1
    return count

# Fungsi untuk menentukan apakah puzzle solveable atau tidak
def IsSolveable(mtrx) :

    sum = 0
    print("[]======================[]")
    print("||",    "i".rjust(5), "||".rjust(5), " kurang".rjust(5),
"||".rjust(3))
    print("[]======================[]")
    for i in range (1,17):
        kurang = Kurang(mtrx, i)

        print("||" , str(i).rjust(5), "||".rjust(5) , str(kurang).rjust(5),
"||".rjust(5))
        sum+=kurang
    posisi = (Position(mtrx, 16)[0]* 4) + Position(mtrx, 16)[1] + 1


    # 2,4,5,7,10,12,13,15
    print("[]======================[]")
    print("Nilai Sigma(Kurang(i)) + X : " , sum + (posisi in
[2,4,5,7,10,12,13,15]))
    if ((sum + (posisi in [2,4,5,7,10,12,13,15])) %2 == 0 ):
        return True
    else:
        return False
```

```
# Fungsi untuk menghitung cost
def Cost(mtrx, depth):

    if (finished(mtrx)):
        return 0
    else:
        count = 0
        num = 1
        for x in mtrx:
            for y in x:
                if (y != 16 and y != num):
                    count+=1
                num+=1

        return count+depth

# Fungsi yang mengembalikan matriks yang tile #16 dimove ke atas
def GenerateNodes(currentNode, antrian, currentId, dict):

    # ParentId, currentId,  Matrix, Cost, Level, char
    posisi = Position(currentNode[2], 16)
    depth = currentNode[4] + 1

    tempCost = depth
    if (posisi[0] != 0) and currentNode[-1] != 'd':         # Up
        temp = Up(currentNode[2])
        tempCost = Cost(temp,depth)

        if (toString(temp) not in dict):
            antrian.append([currentNode[1], currentId, temp , tempCost ,
depth, 'u'])
            dict[toString(temp)] = 'u'
            currentId+=1

    if (posisi[1] != 0)  and currentNode[-1] != 'r':        # Left
        temp = Left(currentNode[2])
        tempCost = Cost(temp,depth)

        if (toString(temp) not in dict):
            antrian.append([currentNode[1], currentId, temp ,tempCost, depth,
'l'])
            dict[toString(temp)] = 'l'
            currentId+=1

    if (posisi[0] != 3)  and currentNode[-1] != 'u':        # Down
        temp = Down(currentNode[2])
        tempCost = Cost(temp,depth)
```

```
        if (toString(temp)not in dict):
            antrian.append([currentNode[1], currentId, temp , tempCost, depth,
'd'])
            dict[toString(temp)] = 'd'
            currentId+=1

    if (posisi[1] != 3) and  currentNode[-1] != 'l':        # Right
        temp = Right(currentNode[2])
        tempCost = Cost(temp,depth)

        if (toString(temp) not in dict):
            antrian.append([currentNode[1], currentId, temp ,tempCost, depth,
'r'])
            dict[toString(temp)] = 'r'
            currentId+=1

    return currentId

def finished(mtrx):
    idx = 1
    for x in mtrx:
        for y  in x:
            if (y != idx):
                return False
            idx+=1
    return True

def getCost(antrian):
    return antrian[3];

def solution(currentNode, visited):
    result = []

    while (currentNode[0] != 0):
        result.append(currentNode[-1])
        currentNode = visited[str(currentNode[0])]

    result.append(currentNode[-1])
    return result[::-1]

def printSolution(mtrx, solution):

    print("Initial")
    Output(mtrx)
    i = 1
    listOfMove = []
    for x in solution:
        if x == 'u':
```

```
            mtrx = Up(mtrx)
            move = 'UP'
        elif x == 'l':
            mtrx = Left(mtrx)
            move = 'LEFT'
        elif x == 'd':
            mtrx = Down(mtrx)
            move = 'DOWN'
        elif x == 'r':
            mtrx = Right(mtrx)
            move = 'RIGHT'

        listOfMove.append(move)
        print("\nMove", i,":", move)
        Output(mtrx)
        i+=1

    print("\nMove yang diperlukan(%d) :" %(len(solution)) , end = " ")
    print(*listOfMove,sep = ", ")


def solve(mtrx):
    nodes = [[0,0,mtrx, 0, 0, '-']]
    visited = {}
    banyakSimpul= 1
    idx = 0

    simpul = { toString(mtrx) : 'CHECK'}

    while (not finished(nodes[0][2])):
        currentNode = nodes.pop(0)
        visited[str(currentNode[1])] = currentNode

        banyakSimpul =  GenerateNodes(currentNode, nodes, banyakSimpul,
simpul)
        idx+=1
        nodes.sort(key = getCost)
    return nodes[0], visited, banyakSimpul


def readFile():

    FileFound = False
    while (not FileFound):
        file = str(input("Nama File : "))
        file = "../test/" + file
        try:
            mtrx = open(file,"r")
            FileFound = True
```

```
        except:
            print("File Tidak Ditemukan")

    puzzle=[] # Menyimpan matrix
    for line in mtrx.readlines():
        puzzle.append( [ int (x) for x in line.split(' ') ] )
    return puzzle
```

2. main.py

```
from solver import *
import time

print("Pilih Jenis Masukkan:")
print("1. File")
print("2. Randomize")

pilihan = int(input("Pilihan : "))

while (pilihan != 1 and pilihan != 2):
    print("Pilihan tidak sesuai, ketikkan angka 1 atau 2")
    pilihan = int(input("Pilihan (Int)>> "))

if (pilihan == 1):
    mtrx = readFile()
else:
    mtrx = RandomizeMatrix()

print("===== Matriks =====")
Output(mtrx)

if (not IsSolveable(mtrx)) :
    print("Puzzle tidak dapat diselesaikan")
else:
    startTime = time.time()

    nodes = []
    visited = {}
    nodes, visited, banyakSimpul  = solve(mtrx)

    endTime = time.time()

    solusi = solution(nodes, visited)
    print("\n==== Urutan Penyelesain Matriks ====")
    printSolution(mtrx, solusi)
    print("Jumlah simpul yang dibangkitkan :",banyakSimpul)
    print("Waktu Eksekusi Program :  %s seconds"  % (endTime - startTime))
```

HASIL EKSEKUSI PROGRAM

1. solveable1.txt

```
Pilih Jenis Masukkan:
1. File
2. Randomize
Pilihan : 1
Nama File : solveable1.txt
===== Matriks =====
   1   2  12   3
   5   6   7   4
   9   8  10  15
  13  14   -  11
[]====================[]
||      i     ||  kurang  ||
[]====================[]
||      1     ||     0    ||
||      2     ||     0    ||
||      3     ||     0    ||
||      4     ||     0    ||
||      5     ||     1    ||
||      6     ||     1    ||
||      7     ||     1    ||
||      8     ||     0    ||
||      9     ||     1    ||
||     10     ||     0    ||
||     11     ||     0    ||
||     12     ||     9    ||
||     13     ||     1    ||
||     14     ||     1    ||
||     15     ||     3    ||
||     16     ||     1    ||
[]====================[]
Nilai Sigma(Kurang(i)) + X :  20
```

```
==== Urutan Penyelesain Matriks ====
Initial
   1   2  12   3
   5   6   7   4
   9   8  10  15
  13  14   -  11

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Move 16 : DOWN
   1   2   3   4
   5   6   7   8
   9   -  10  12
  13  14  11  15

Move 17 : RIGHT
   1   2   3   4
   5   6   7   8
   9  10   -  12
  13  14  11  15

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

Move 19 : RIGHT
   1   2   3   4
   5   6   7   8
   9  10  11  12
  13  14  15   -
```

```
Move yang diperlukan(19) : UP, LEFT, UP, RIGHT, UP, RIGHT, DOWN, LEFT, DOWN, DOWN, RIGHT, UP, UP, LEFT, LEFT, DOWN, RIGHT, DOWN, RIGHT
Jumlah simpul yang dibangkitkan : 7142
Waktu Eksekusi Program :  0.49808216094970703 seconds
```

2. solveable2.txt

```
Pilih Jenis Masukkan:
1. File
2. Randomize
Pilihan : 1
Nama File : solveable2.txt
===== Matriks =====
   6   5   2   4
   9   1   3   8
  10   -   7  15
  13  14  12  11
[]====================[]
||     i    ||  kurang  ||
[]====================[]
||     1    ||     0    ||
||     2    ||     1    ||
||     3    ||     0    ||
||     4    ||     2    ||
||     5    ||     4    ||
||     6    ||     5    ||
||     7    ||     0    ||
||     8    ||     1    ||
||     9    ||     4    ||
||    10    ||     1    ||
||    11    ||     0    ||
||    12    ||     1    ||
||    13    ||     2    ||
||    14    ||     2    ||
||    15    ||     4    ||
||    16    ||     6    ||
[]====================[]
Nilai Sigma(Kurang(i)) + X :  34
```

```
==== Urutan Penyelesain Matriks ====
Initial
   6   5   2   4
   9   1   3   8
  10   -   7  15
  13  14  12  11

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

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

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

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

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

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

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

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

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

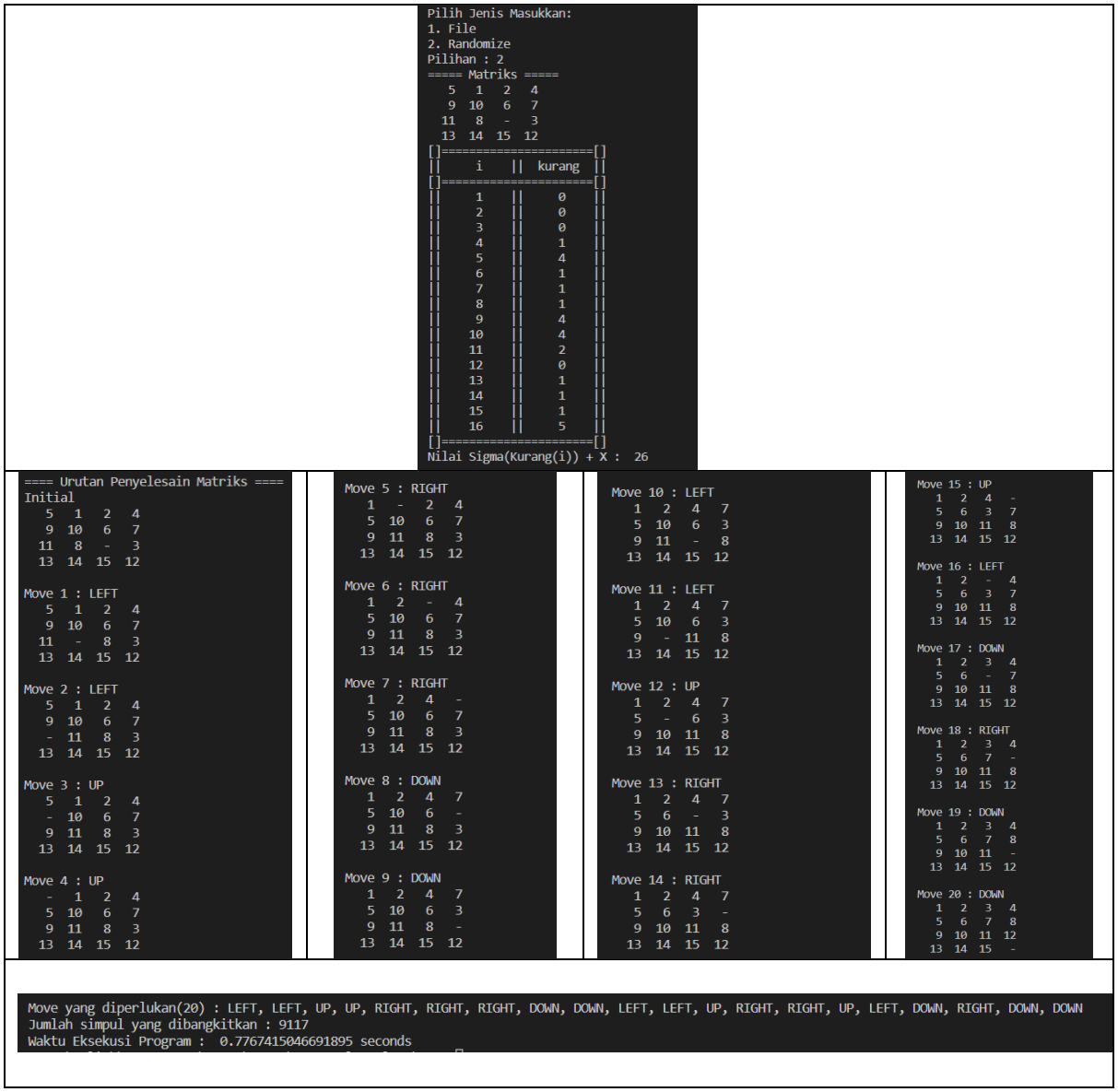
```
Move 10 : DOWN
   1   2   3   4
   5   6   -   8
   9  10   7  15
  13  14  12  11

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

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

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

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

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

Move 16 : DOWN
   1   2   3   4
   5   6   7   8
   9  10  11  12
  13  14   -  15

Move 17 : RIGHT
   1   2   3   4
   5   6   7   8
   9  10  11  12
  13  14  15   -
```

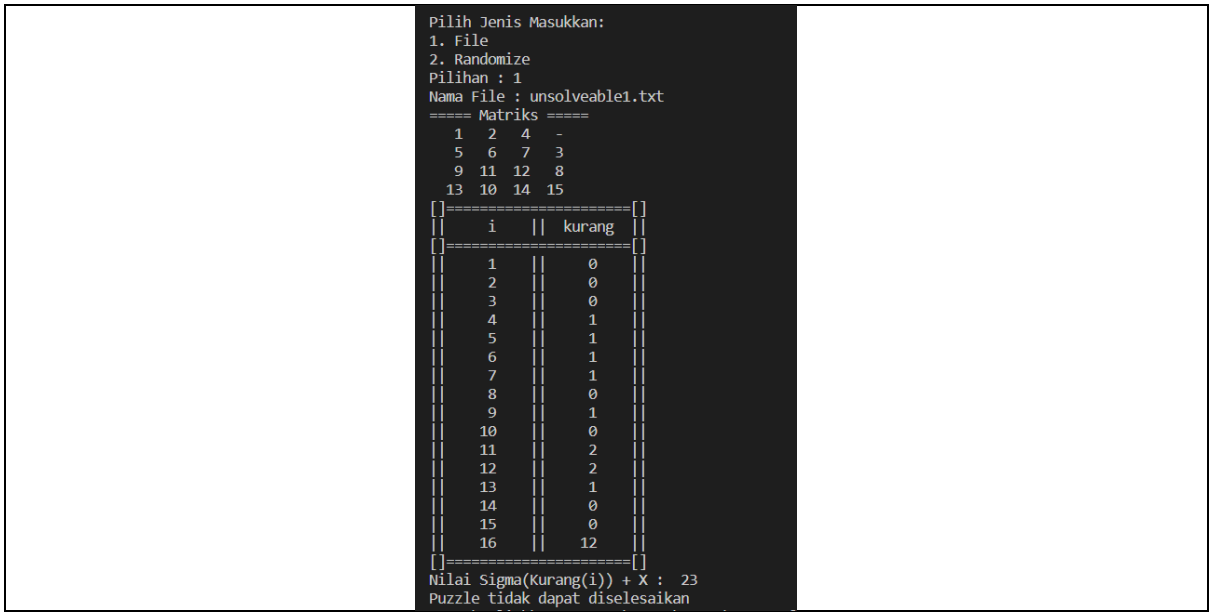
```
Move yang diperlukan(17) : LEFT, UP, UP, RIGHT, DOWN, LEFT, UP, RIGHT, RIGHT, DOWN, DOWN, DOWN, RIGHT, UP, LEFT, DOWN, RIGHT
Jumlah simpul yang dibangkitkan : 1080
Waktu Eksekusi Program :  0.02989053726196289 seconds
```

3. solveable3.txt

Puzzle awalnya digenerate secara random, dan disimpan secara manual ke dalam file solveable3.txt untuk dokumentasi.

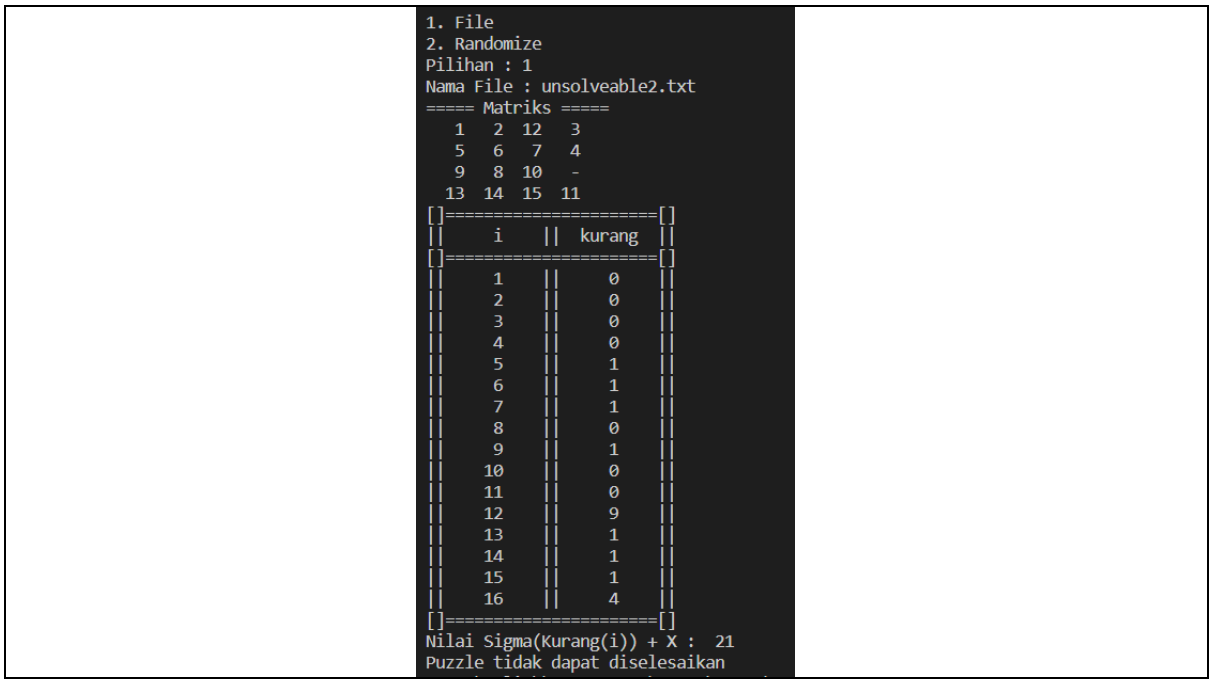


4. unsolveable1.txt



```
Pilih Jenis Masukkan:
1. File
2. Randomize
Pilihan : 1
Nama File : unsolveable1.txt
===== Matriks =====
   1   2   4   -
   5   6   7   3
   9  11  12   8
  13  10  14  15
[]====================[]
||     i    ||  kurang  ||
[]====================[]
||     1    ||     0    ||
||     2    ||     0    ||
||     3    ||     0    ||
||     4    ||     1    ||
||     5    ||     1    ||
||     6    ||     1    ||
||     7    ||     1    ||
||     8    ||     0    ||
||     9    ||     1    ||
||    10    ||     0    ||
||    11    ||     2    ||
||    12    ||     2    ||
||    13    ||     1    ||
||    14    ||     0    ||
||    15    ||     0    ||
||    16    ||    12    ||
[]====================[]
Nilai Sigma(Kurang(i)) + X :  23
Puzzle tidak dapat diselesaikan
```

5. unsolveable2.txt



```
1. File
2. Randomize
Pilihan : 1
Nama File : unsolveable2.txt
===== Matriks =====
   1   2  12   3
   5   6   7   4
   9   8  10   -
  13  14  15  11
[]====================[]
||     i    ||  kurang  ||
[]====================[]
||     1    ||     0    ||
||     2    ||     0    ||
||     3    ||     0    ||
||     4    ||     0    ||
||     5    ||     1    ||
||     6    ||     1    ||
||     7    ||     1    ||
||     8    ||     0    ||
||     9    ||     1    ||
||    10    ||     0    ||
||    11    ||     0    ||
||    12    ||     9    ||
||    13    ||     1    ||
||    14    ||     1    ||
||    15    ||     1    ||
||    16    ||     4    ||
[]====================[]
Nilai Sigma(Kurang(i)) + X :  21
Puzzle tidak dapat diselesaikan
```

SOURCE CODE

https://github.com/SurTan02/Tucil3_13520059

LAMPIRAN

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		√