LAPORAN TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA

PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA BRANCH AND BOUND

Oleh :

Ilham Prasetyo Wibowo 13520013

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

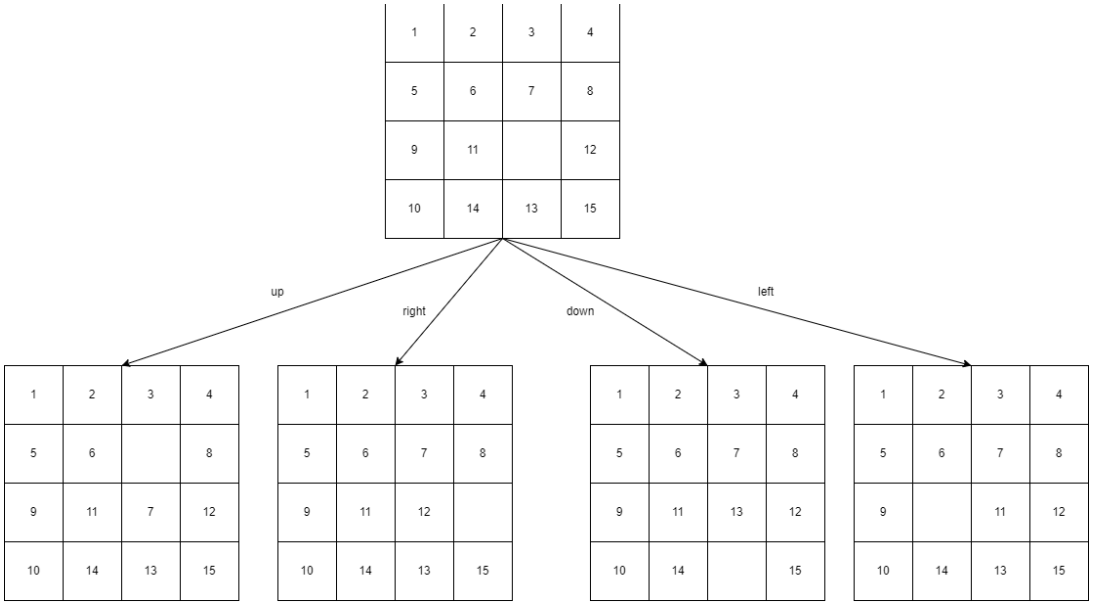
A. DESKRIPSI ALGORITMA

Algoritma Branch and Bound digunakan untuk persoalan optimasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan persoalan. Algortima branch and bound merupakan gabungan dari algortima pencarian melebar (BFS) dan least cost search. Algoritma brach and bound secara umum sebagai berikut.

1. Setiap simpul diberi sebuah nilai cost.
2. Simpul berikutnya yang akan diexpand adalah simpul yang memiliki cost terkecil (minimisasi).

Algortima penyelesaian 15-puzzle dimulai dengan menghitung nilai jumlah kurang(i) + x untuk mengetahui apakah matriks awal bisa menuju solusi. Nilai kurang(i) adalah banyak ubin yang nilainya kurang dari i, tetapi memiliki posisi lebih dari posisi i. Kemudian dijumlahkan dan ditambahkan nilai x. X memiliki nilai 0 atau 1.Jika nilai jumlah kurang adalah ganjil, maka tidak ada solusi, program akan mengeluarkan pesan.

Program menggunakan sebuah kelas priorityQueue dan node. Selama priority queue tidak kosong dan solusi belum ditemukan, program akan membangkitkan simpul baru(node). Simpul simpul tersebut adalah hasil matriks sebelumnya yang ubin kosong sudah dipindahkan ke kiri, atas, bawah, atau kanan.



Tiap simpul memiliki sebuah cost. Cost terbagi menjadi dua yaitu fcost yang merepresantikan kedalaman simpul, dan gcost yang menghitung ubin yang tidak sesuai dengan konfigurasi akhir dari matriks. Sebelum node dimasukkan ke dalam priority queue, dicek terlebih dahulu apakah node bisa diselesaikan. Jika jumlah kurang adalah genap, maka node akan dimasukkan ke dalam priority queue. Iterasi selanjutnya akan mengambil node dengan nilai cost terkecil dari priority queue. Langkah ini terus dilakukan sampai solusi ditemukan atau queue kosong.

B. SOURCE CODE

priorityQueue.py

```
class priorityQueue:
    def __init__(self):
        self.queue = []

    def push(self, node):
        self.queue.append(node)

    def pop(self):
        minidx = 0
        for i in range(len(self.queue)): #ambil cost
            if (self.queue[i].cost < self.queue[minidx].cost):
                minidx = i
        min = self.queue[minidx]
        del self.queue[minidx]
        return min

    def empty(self):
        if not self.queue:
            return True
        else:
            return False
    def printqueuecost(self):
        for i in range(len(self.queue)):
            print(self.queue[i].cost, end=" ")
        print()
```

node.py

```
#nodes

class node:

    def __init__(self, parent, mat, fcost, cost,move):
        self.parent = parent
        self.mat = mat
        self.fcost = fcost
        self.cost = cost
        self.move = move
```

main.py

```
#asumsi tile kosong adalah ubin dengan nilai 16
import timeit
import numpy as np
```

```python
import random
from priorityQueue import priorityQueue
from node import node
EMPTY = 16

def newNode(mat, i,j,k,l, parent,target,move):
    new_mat = copyMatrix(mat)
    swap(new_mat,i,j,k,l)
    #printMatrix(new_mat)
    #new_mat[i][j], new_mat[k][l] = new_mat[k][l], new_mat[i][j]
    fcost = parent.fcost + 1
    cost = fcost + gcost(new_mat,target)

    new_node = node(parent, new_mat,fcost, cost,move)
    return new_node

#read from file
def read_file(filename):
    with open(filename, 'r') as f:
        l = [[int(num) for num in line.split(' ')] for line in f]
    return l

#Copy matrix, return matriks hasil copy
def copyMatrix(mat2):
    mat1 = [[0 for i in range(4)] for j in range(4)]
    for i in range(len(mat1)):
        for j in range(len(mat1[0])):
            mat1[i][j] = mat2[i][j]
    return mat1

#menghitung cost g(P) menuju target
def gcost(mat, targetmat):
    count = 0
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            if (mat[i][j] != targetmat[i][j]) :
                count += 1

    return count

#mengembalikan posisi bilangan n pada matrix
def posisi(mat,n):
    for i in range(4):
        for j in range(4):
            if (mat[i][j] == n):
                return i,j

    return -1,-1
```

```
#fungsi KURANG(i)
def kurang(mat,num):
    ipos,jpos = posisi(mat,num)
    countkurang = 0
    for i in range(ipos,len(mat)):
        if (i == ipos) :
            for j in range(jpos,len(mat[0])):
                if (mat[i][j] < num) :
                    countkurang += 1
        else :
            for j in range(len(mat[0])):
                if (mat[i][j] < num) :
                    countkurang += 1
    return countkurang

#Print solusi
def printPath(node):

    if node == None:
        return

    printPath(node.parent)
    printmove(node.move)
    printMatrix(node.mat)
    print()

#Sumkurang untuk mengetahui sebuah matriks reachable goal atau bukan
def sumkurang(mat):
    sum = 0
    ckr = 0
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            ckr = kurang(mat,mat[i][j])
            #print("KURANG("+str(mat[i][j])+") = " + str(ckr))
            sum += ckr
    #print(sum)
    sum += countX(mat)
    #print("JUMLAH KURANG(i) + X = " + str(sum))
    return sum

def printkurang(mat):
    sum = 0
    ckr = 0
    arr = [0 for i in range(16)]
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            ckr = kurang(mat,mat[i][j])
```

```
            arr[mat[i][j]-1] = ckr
            sum += ckr

    for i in range(16):
        print("KURANG("+str(i+1)+") = " + str(arr[i]))
    sum += countX(mat)
    print("JUMLAH KURANG(i) + X = " + str(sum))

#menentukan nilai x, 0 atau 1
def countX(mat):
    ipos,jpos = posisi(mat,EMPTY)
    if (ipos%2 == 0) :
        if (jpos %2 == 0) :
            return 0
        else :
            return 1
    else :
        if (jpos%2 == 0) :
            return 1
        else :
            return 0

#Swap matriks dengan posisi mat[i][j] dengan mat[k][l]
def swap(mat, i,j, k,l):
    temp = mat[k][l]
    mat[k][l] = mat[i][j]
    mat[i][j] = temp

#print matris
def printMatrix(mat):
    for i in range(len(mat)):
        print("---------------------")
        for j in range(len(mat[0])):
            if (j == 0) :
                    print("| ", end="")
            if (mat[i][j] == EMPTY):
                print(" ", end = "  | ")
            elif (mat[i][j] < 10) :
                print(mat[i][j], end="  | ")
            else :
                print(mat[i][j], end = " | ")

        print()
    print("---------------------")

#Cek apakah matriks sudah mencapai solusi
#Mat dan target memiliki ukuran sama
def solution(mat,target):
```

```
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            if (mat[i][j] != target[i][j]) :
                return False
    return True

def safe(a,b):
    if (b == -1):
        return True
    elif (a == 1):
        if (b == 3):
            return False
        else :
            return True
    elif (a == 0):
        if (b == 2):
            return False
        else :
            return True
    elif (a == 2):
        if (b == 0):
            return False
        else :
            return True
    elif(a == 3):
        if (b == 1):
            return False
        else:
            return True

def printmove(n):
    if (n != -1):
        print("||||||||||||||||||||")
        if (n == 0):
            print("^^^^^ MOVE UP ^^^^^")
        elif (n == 1):
            print(">>>>> MOVE RIGHT >>>>>")
        elif (n == 2):
            print("vvvvv MOVE DOWN vvvvv")
        elif (n == 3):
            print("<<<<< MOVE LEFT <<<<<")
        print("||||||||||||||||||||")

#solve
def solve(mat,target):
    start = timeit.default_timer()
    pq = priorityQueue()
    gc = gcost(mat,target)
```

```
    found = False
    root = node(None,mat,0,gc,-1)
    pq.push(root)
    jumlahsimpul = 1

    if (sumkurang(mat) %2 != 0) :
        print("SOLUSI TIDAK DITEMUKAN")
    else :
        row = [-1,0,1,0]
        col = [0,1,0,-1]
        while (not pq.empty()):
            next = pq.pop()

            if (solution(next.mat, target)):
                print("SOLUSI YANG DITEMUKAN : ")
                printPath(next)
                found = True
                break
            ipos,jpos = posisi(next.mat,EMPTY)
            #generate kemungkinan matriks
            for i in range(4):
                #mengecek jika move terakhir up maka move selanjutnya != down,
left != right dan sebaliknya
                if (safe(i,next.move)):
                    #cek apakah melebihi batas
                    if (ipos+row[i] >= 0 and ipos+row[i] < len(mat) and
jpos+col[i] >= 0 and jpos+col[i] < len(mat[0])):
                        new_node =
newNode(next.mat,ipos,jpos,ipos+row[i],jpos+col[i],next,target,i)
                        #tidak dimasukkan queue jika tidak menuju solusi
                        if (sumkurang(next.mat)%2 == 0):
                            pq.push(new_node)
                        jumlahsimpul += 1
        if (not found):
            print("SOLUSI TIDAK DITEMUKAN")
    end = timeit.default_timer()
    print("Waktu eksekusi program : " + str(end-start))
    print("Jumlah simpul yang dibangkitkan : " + str(jumlahsimpul))

def createTarget(n):
    targetmats = [[0 for i in range(n)] for j in range(n)]
    num34 = 1
    for i in range(n):
        for j in range(n):
            targetmats[i][j] = num34
            num34 += 1
    return targetmats
```

```
#solvable matrix
#matriks target diacak
def createRandomMatrix(diff,target):
    mtx = copyMatrix(target)
    row = [-1,0,1,0]
    col = [0,1,0,-1]
    i = 0
    while(i != diff):
        ipos,jpos = posisi(mtx,EMPTY)
        x = random.randint(0,3)
        if (ipos+row[x] >= 0 and ipos+row[x] < len(mtx) and jpos+col[x] >= 0
and jpos+col[x] < len(mtx[0])):
            swap(mtx,ipos,jpos,ipos+row[x],jpos+col[x])
            i += 1
    return mtx


#TESTING
finalmat = createTarget(4)
print("SELAMAT DATANG DI 15-PUZZLE SOLVER")
print("1. BANGKITKAN MATRIKS ACAK")
print("2. PILIH MATRIKS DARI MASUKAN FILE")
print("3. KELUAR")
choice = int(input("PILIHAN : "))
while (not choice == 3):
    if (choice == 1):
        difficulty = int(input("MASUKKAN JUMLAH PENGACAKAN : "))
        mats = createRandomMatrix(difficulty, finalmat)
        print("MATRIKS AWAL : ")
        printMatrix(mats)
        printkurang(mats)
        mat = np.matrix(mats)
        solve(mats,finalmat)

    elif (choice == 2):
        filename = input("MASUKKAN NAMA FILE : ")
        mats = read_file(filename)
        print("MATRIKS AWAL : ")
        printMatrix(mats)
        printkurang(mats)
        solve(mats,finalmat)

    else:
        print("MASUKAN TIDAK VALID!")
    print("1. BANGKITKAN MATRIKS ACAK")
    print("2. PILIH MATRIKS DARI MASUKAN FILE")
    print("3. KELUAR")
    choice = int(input("PILIHAN : "))
```

C. HASIL PENGUJIAN

Lima matriks yang akan digunakan dalam pengujian adalah sebagai berikut.

input1.txt
1 6 2 4
5 15 3 11
16 9 8 7
13 10 14 12

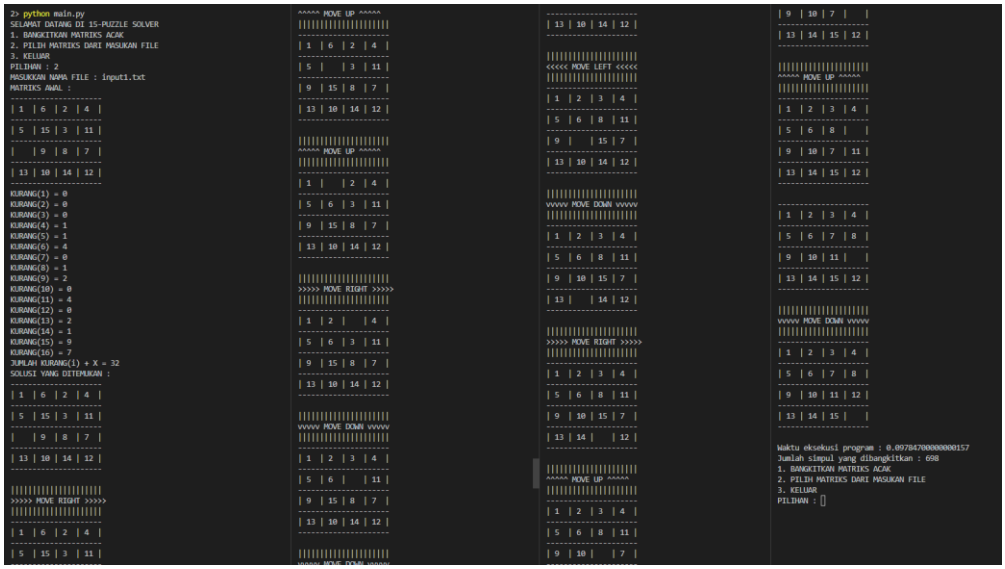
input2.txt
2 16 3 4
1 6 7 8
5 9 11 12
13 10 14 15

input3.txt
2 3 7 4
1 5 15 8
13 6 16 11
10 9 14 12

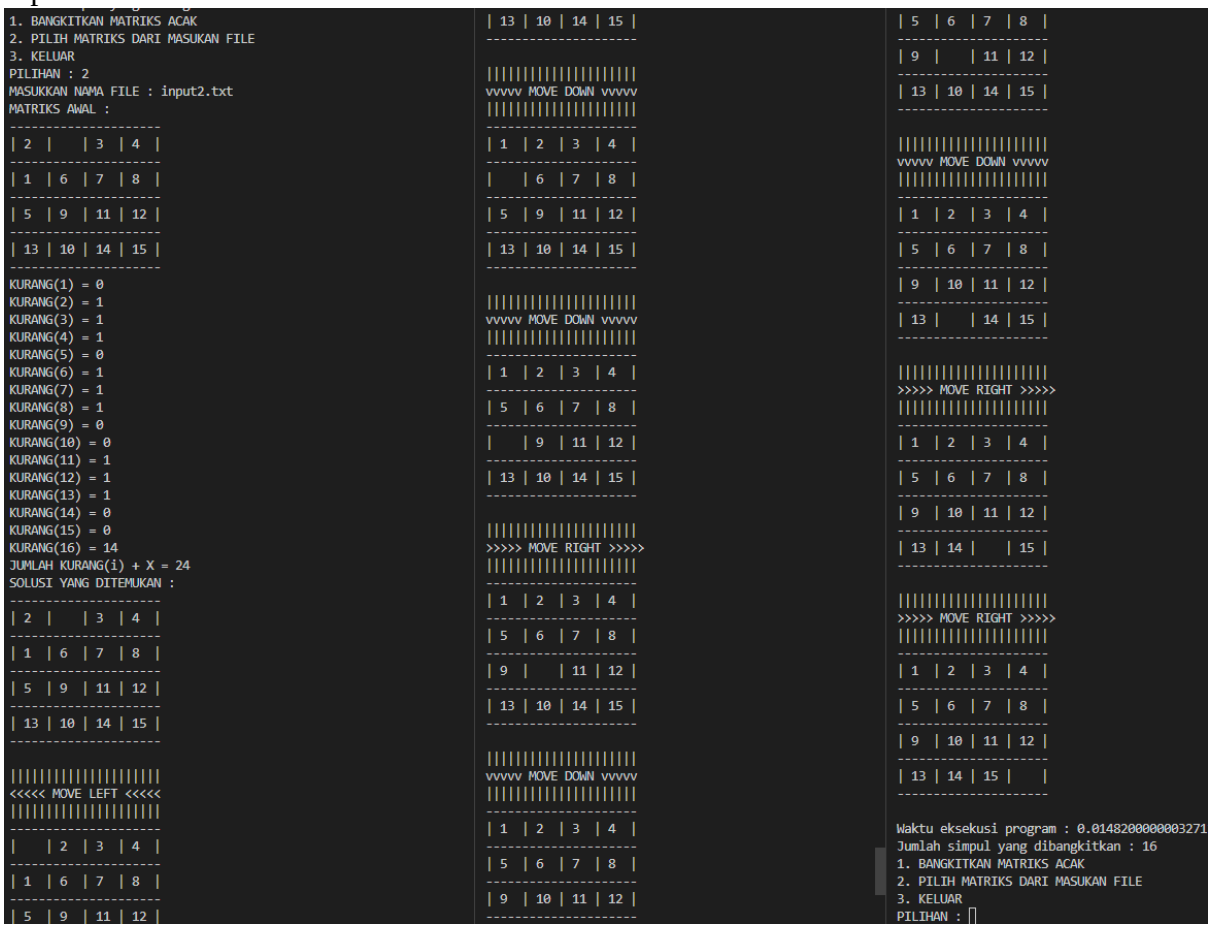
input4.txt
10 5 9 7
14 15 1 6
3 4 12 16
13 8 11 2

input5.txt
2 3 11 4
1 6 10 8
9 5 12 15
13 14 16 7

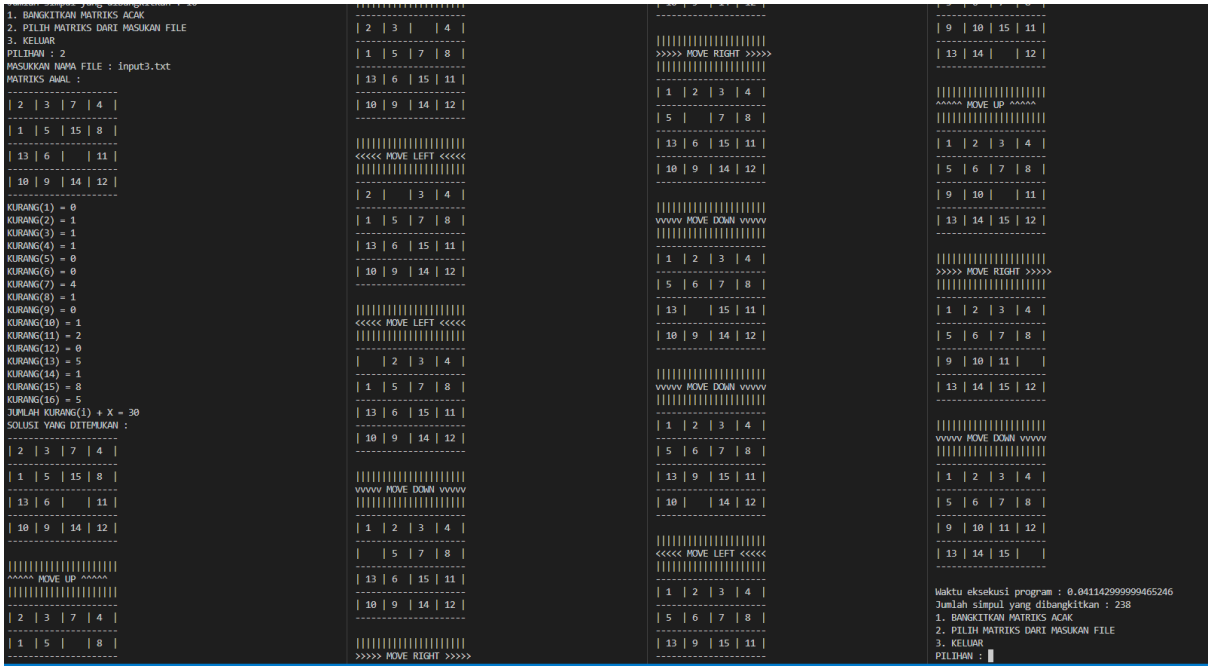
Tangkapan Layar hasil uji :
Input1 :



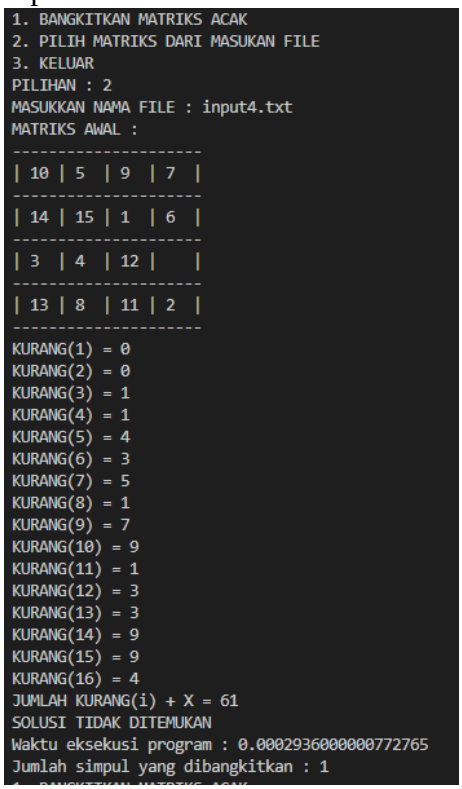
Input2:



Input3:



Input4:



Input5:

```
1. BANGKITKAN MATRIKS ACAK
2. PILIH MATRIKS DARI MASUKAN FILE
3. KELUAR
PILIHAN : 2
MASUKKAN NAMA FILE : input5.txt
MATRIKS AWAL :
--------------------
| 10 | 13 | 5  | 8  |
--------------------
| 11 | 3  |    | 1  |
--------------------
| 12 | 4  | 14 | 9  |
--------------------
| 2  | 6  | 15 | 7  |
--------------------
KURANG(1) = 0
KURANG(2) = 0
KURANG(3) = 2
KURANG(4) = 1
KURANG(5) = 4
KURANG(6) = 0
KURANG(7) = 0
KURANG(8) = 6
KURANG(9) = 3
KURANG(10) = 9
KURANG(11) = 7
KURANG(12) = 5
KURANG(13) = 11
KURANG(14) = 4
KURANG(15) = 1
KURANG(16) = 9
JUMLAH KURANG(i) + X = 63
SOLUSI TIDAK DITEMUKAN
Waktu eksekusi program : 0.00025740000000951113
Jumlah simpul yang dibangkitkan : 1
```

D. TAUTAN MENUJU DRIVE PROGRAM

https://github.com/ilhamwibowo/Tugas-Kecil-3-STIMA

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		✓