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

Fikri Ihsan Fadhiilah (13520148)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

13520148@std.stei.itb.ac.id

Cara Kerja Program Branch and Bound

1. Pengecekan apakah kondisi Akhir dapat tercapai atau tidak dengan menggunakan fungsi kurang

   $\sum_{i=1}^{16} KURANG(i) + X$

   Jika bernilai genap maka dapat diselesaikan, jika ganjil tidak dapat diselesaikan.

```
def is_have_solution(arr_for_checking,black_tile):
    que=[0 for i in range (16)]
    sum=0
    for i in range(np.size(arr_for_checking)):
        #component=0
        if arr_for_checking[i]==16 and i in black_tile:
            sum+=1
        for j in range(i+1,np.size(arr_for_checking)):
            if arr_for_checking[i]>arr_for_checking[j]:
                sum+=1
                que[arr_for_checking[i]-1]+=1
                #component+=1
        #print("Nilai fungsi Kurang("+str(arr_for_checking[i])+") = " + str(component))
    for i in range(16):
        print("Nilai fungsi Kurang("+str(i+1)+") = " + str(que[i]))
    print("\nTotal nilai Fungsi KURANG(i) + X adalah " + str(sum)+"\n")
    if sum%2==0:
        return True
    else:
        return False
```

2. Memasukkan sebuah simpul ke simpul hidup selama tujuan belum tercapai

```
    if next_node.x != 0:
        moved = move_up(deepcopy(next_node.state),next_node.x,next_node.y)
        if tuple(np.reshape(moved,16)) not in visited: #Jika susunan puzzle belum
            move(que_,moved,next_node)
            node_generated+=1 #Jumlah simpul yang dibangkitkan
            visited.add(tuple(np.reshape(moved,16))) #Me-list semua susunan puzzle
    if next_node.y != 3:
        moved = move_right(deepcopy(next_node.state),next_node.x,next_node.y)
        if tuple(np.reshape(moved,16)) not in visited:
            move(que_,moved,next_node)
            node_generated+=1
            visited.add(tuple(np.reshape(moved,16)))
    if next_node.x != 3:
        moved = move_down(deepcopy(next_node.state),next_node.x,next_node.y)
        if tuple(np.reshape(moved,16)) not in visited:
            move(que_,moved,next_node)
            node_generated+=1
            visited.add(tuple(np.reshape(moved,16)))
    if next_node.y !=0:
        moved = move_left(deepcopy(next_node.state),next_node.x,next_node.y)
        if tuple(np.reshape(moved,16)) not in visited:
            move(que_,moved,next_node)
            node_generated+=1
            visited.add(tuple(np.reshape(moved,16)))
    next_node=que_.pop(0) #Mengambil node dengan cost terkecil
sol=next_node #Menambahkan solusi
```

3. Mengurutkan simpul hidup berdasarkan cost lalu jika sama berdasarkan kedalaman setiap kali ada simpul yang di tambahkan ke simpul hidup

```
def get_cost(node_):
    return node_.cost,node_.depth

def ins_to_que(que_,node_):
    que_.append(node_)
    que_.sort(key=get_cost)

def move(que_,moved,node_):
    x,y=get_blank_location(moved)
    moved_node = node(moved,node_,node_.depth+1,x,y,count_cost(moved)+node_.depth+1)
    ins_to_que(que_,moved_node)
```

4. Mengambil simpul dengan cost terkecil untuk melanjutkan penelusuran

```
next_node=que_.pop(0)
```

5. Mengulangi Langkah 2-4 sampai tercapai bentuk matrix Akhir
6. Memasukkan node ke list jawaban

```
sol=next_node
```

7. Menghapus seluruh antrian simpul hidup

```
que_.clear() #Menghapus semua antrian simpul hidup
```

8. Menampilkan jawaban beserta urutan pergerakannya dari bentuk awal

SOURCE PROGRAM

```
import numpy as np
from copy import deepcopy
import time

def is_have_solution(arr_for_checking,black_tile):
    '''
    Mengecek apakah puzzle memiliki penyelesaian atau tidak berdasarkan fungsi KURANG(i)
    '''
    que=[0 for i in range (16)]
    sum=0
    for i in range(np.size(arr_for_checking)):
        if arr_for_checking[i]==16 and i in black_tile:
            sum+=1
        for j in range(i+1,np.size(arr_for_checking)):
            if arr_for_checking[i]>arr_for_checking[j]:
                sum+=1
                que[arr_for_checking[i]-1]+=1 #Untuk Keperluan display fungsi KURANG(i) dari setiap ubin tidak kosong

    #Keperluan Display
    for i in range(16):
        print("Nilai fungsi Kurang("+str(i+1)+") = " + str(que[i]))
    print("\nTotal nilai Fungsi KURANG(i) + X adalah " + str(sum)+"\n")
    #Keperluan Display

    if sum%2==0:
        return True
    else:
        return False

def display_matrix(matrix):
    for i in range(4):
        for j in range(4):
            if matrix[i][j] == 16:
                print(" \t",end="")
            else:
                print(str(matrix[i][j])+"\t",end="")
        print("")

#Gerak kekanan
def move_right(state,x,y):
    temp = state[x][y]
```

```
    state[x][y]=state[x][y+1]
    state[x][y+1]=temp
    return state
#Gerak keatas
def move_up(state,x,y):
    temp = state[x][y]
    state[x][y]=state[x-1][y]
    state[x-1][y]=temp
    return state
#Gerak kekiri
def move_left(state,x,y):
    temp = state[x][y]
    state[x][y]=state[x][y-1]
    state[x][y-1]=temp
    return state
#Gerak kebawah
def move_down(state,x,y):
    temp = state[x][y]
    state[x][y]=state[x+1][y]
    state[x+1][y]=temp
    return state

#Mendapatkan letak ubin kosong
def get_blank_location(arr):
    for i in range(4):
        for j in range(4):
            if arr[i][j]==16:
                x=i
                y=j
                break
    return x,y

#Untuk menghitung cost dari sebuah simpul
def count_cost(arr):
    cost=0
    arr_for_checking=np.ravel(arr)
    for i in range(np.size(arr_for_checking)):
        if arr_for_checking[i] != 16 and i+1!=arr_for_checking[i]:
            cost+=1
    return cost

#Keperluan sorting simpul hidup
def get_cost(node_):
    return node_.cost,node_.depth
#Keperluan sorting simpul hidup
```

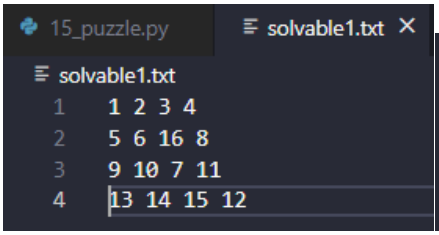
```
def ins_to_que(que_,node_):
    que_.append(node_)
    que_.sort(key=get_cost) #Mengurutkan simpul hidup berdasarkan cost lalu jika
sama, berdasarkan kedalaman simpul

#Menyisipkan simpul simpul yang terbentuk dari setiap pergerakan kedalam simpul
hidup
def move(que_,moved,node_):
    '''
    que_ adalah list simpul hidup
    moved adalah simpul denga bentuk matrix setelah bergerak
    node_ adalah simpul denga bentuk matrix sebelum bergerak

    '''
    x,y=get_blank_location(moved)
    moved_node =
node(moved,node_,node_.depth+1,x,y,count_cost(moved)+node_.depth+1) #Membentuk
object simpul
    ins_to_que(que_,moved_node)

#Pergerakan dari setiap node
def solve(que_,node_,visited):
    next_node=node_
    global node_generated #Untuk menghitung jumlah simpul yang dibangkitkan
    global sol
    while not(np.array_equal(goal_state,next_node.state)): #Hingga menemukan
hasil akhir
        if next_node.x != 0:
            moved = move_up(deepcopy(next_node.state),next_node.x,next_node.y)
            if tuple(np.reshape(moved,16)) not in visited: #Jika susunan puzzle
belum pernah dijumpai maka, dimasukkan ke dalam simpul hidup
                move(que_,moved,next_node)
                node_generated+=1 #Jumlah simpul yang dibangkitkan
                visited.add(tuple(np.reshape(moved,16))) #Me-list semua susunan
puzzle yang pernah dijumpai
        if next_node.y != 3:
            moved = move_right(deepcopy(next_node.state),next_node.x,next_node.y)
            if tuple(np.reshape(moved,16)) not in visited:
                move(que_,moved,next_node)
                node_generated+=1
                visited.add(tuple(np.reshape(moved,16)))
        if next_node.x != 3:
            moved = move_down(deepcopy(next_node.state),next_node.x,next_node.y)
            if tuple(np.reshape(moved,16)) not in visited:
                move(que_,moved,next_node)
```

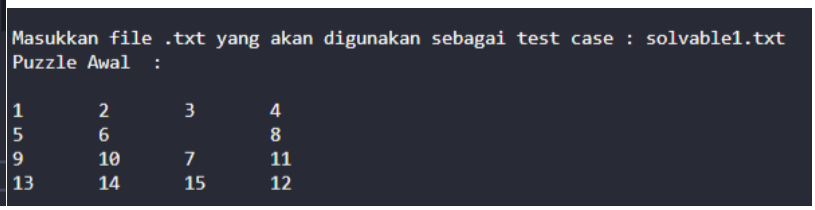
```
                node_generated+=1
                visited.add(tuple(np.reshape(moved,16)))
        if next_node.y !=0:
            moved = move_left(deepcopy(next_node.state),next_node.x,next_node.y)
            if tuple(np.reshape(moved,16)) not in visited:
                move(que_,moved,next_node)
                node_generated+=1
                visited.add(tuple(np.reshape(moved,16)))
        next_node=que_.pop(0) #Mengambil node dengan cost terkecil
    sol=next_node #Menambahkan solusi
    que_.clear() #Menghapus semua antrian simpul hidup

#Menampilkan urutan penyelesaian
def display_path(node_):
    if node_.parents_node != None: #Apabila parents node masih memiliki parents
maka rekursifkan
        display_path(node_.parents_node)
        display_matrix(node_.state)
        print("\n")
    else:
        display_matrix(node_.state)
        print("\n")

#Convert txt menjadi matrix
def teks_to_matriks(_inputfile):
    _case = []
    with open(_inputfile) as file:
        for item in file:
            _case.append([int(i) for i in item.split()])
    return _case

class node(object):
    def
__init__(self,state,parents_node,depth,Xblank_location,Yblank_location,cost):
        self.state=state
        self.parents_node=parents_node
        self.depth=depth
        self.x=Xblank_location
        self.y=Yblank_location
        self.cost=cost

if __name__ == '__main__':
    print("\n\n=== Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and
Bound ===\n")
    goal_state=np.array([
```

```
        [1,2,3,4],
        [5,6,7,8],
        [9,10,11,12],
        [13,14,15,16]
    ])
    input_file= input("\nMasukkan file .txt yang akan digunakan sebagai test case
: ")
    is_= teks_to_matriks("test/"+input_file)

    black_tile=[1,3,4,6,9,11,12,14]
    arr_for_checking=np.ravel(is_) #Mengubah matriks mencjadi 1 dimensi untuk
pengecekan fungsi KURANG(i)
    node_generated = 0 #Total jumlah simpul yang terbentuk

    print("Puzzle Awal  : \n")
    display_matrix(is_)
    print("")

    if is_have_solution(arr_for_checking,black_tile):
        back_to_2d = arr_for_checking.reshape(4,4) #mengembalikan bentuk menjadi
2d
        #initiate root ---------------------------
        urutan=[] #List simpul hidup
        sol=None #Jawaban
        visited=set() #List simpul yang pernah dikunjungi
        x_start,y_start=get_blank_location(back_to_2d)
        start_node = node(back_to_2d,None,0,x_start,y_start,99)
        urutan.append(start_node) #Menambahkan simpul dengan matrix bentuk awal
kedalam simpul hidup
        #initiate root ---------------------------

        #Runtime-----------------------
        start_time = time.time()
        solve(urutan,start_node,visited)
        selesai=time.time()-start_time
        #Runtime-----------------------

        print("\nLangkah Penyelesaian\n")
        display_path(sol)
        print("Jumlah simpul yang dibangkitkan = "+str(node_generated)+"\n")
        print("Total waktu eksekusi penyelesaian : " + str(selesai))


    else:
        print("\nGA BISA DISELESAIIN NICH")
```

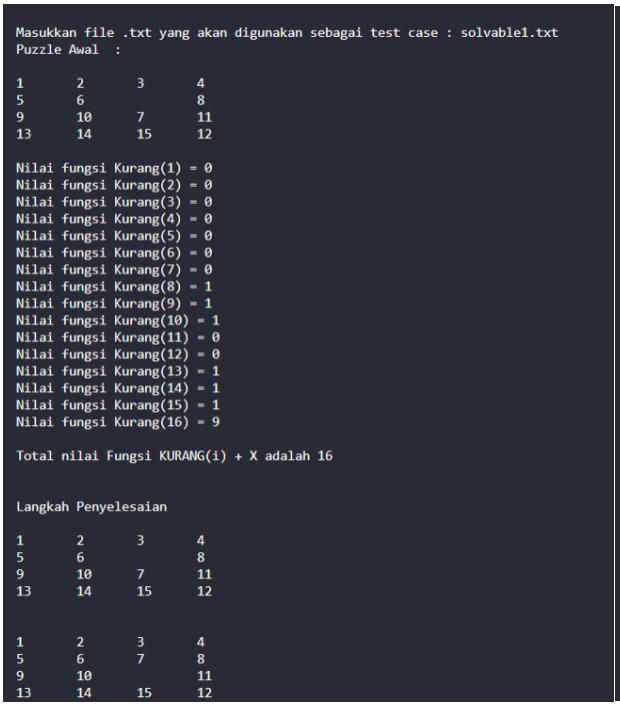
Screenshoot Input-Output Program

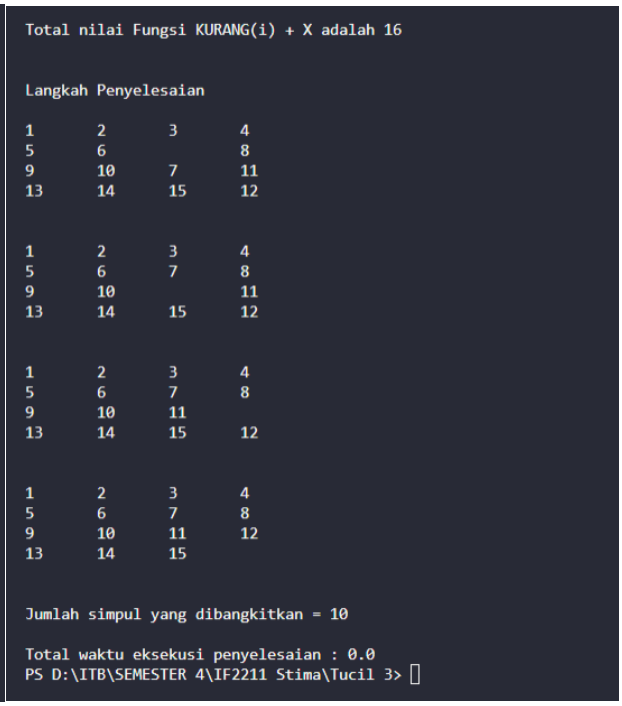
Input :



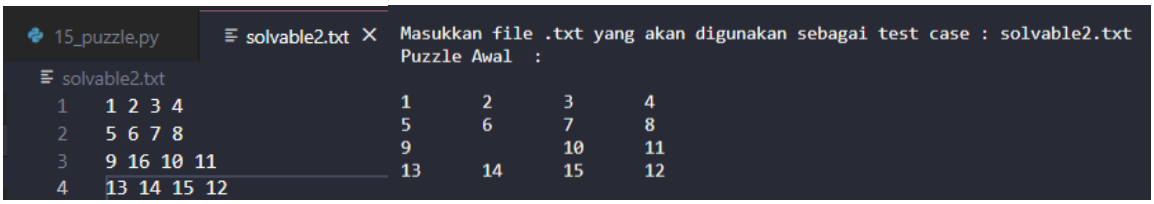


Output:

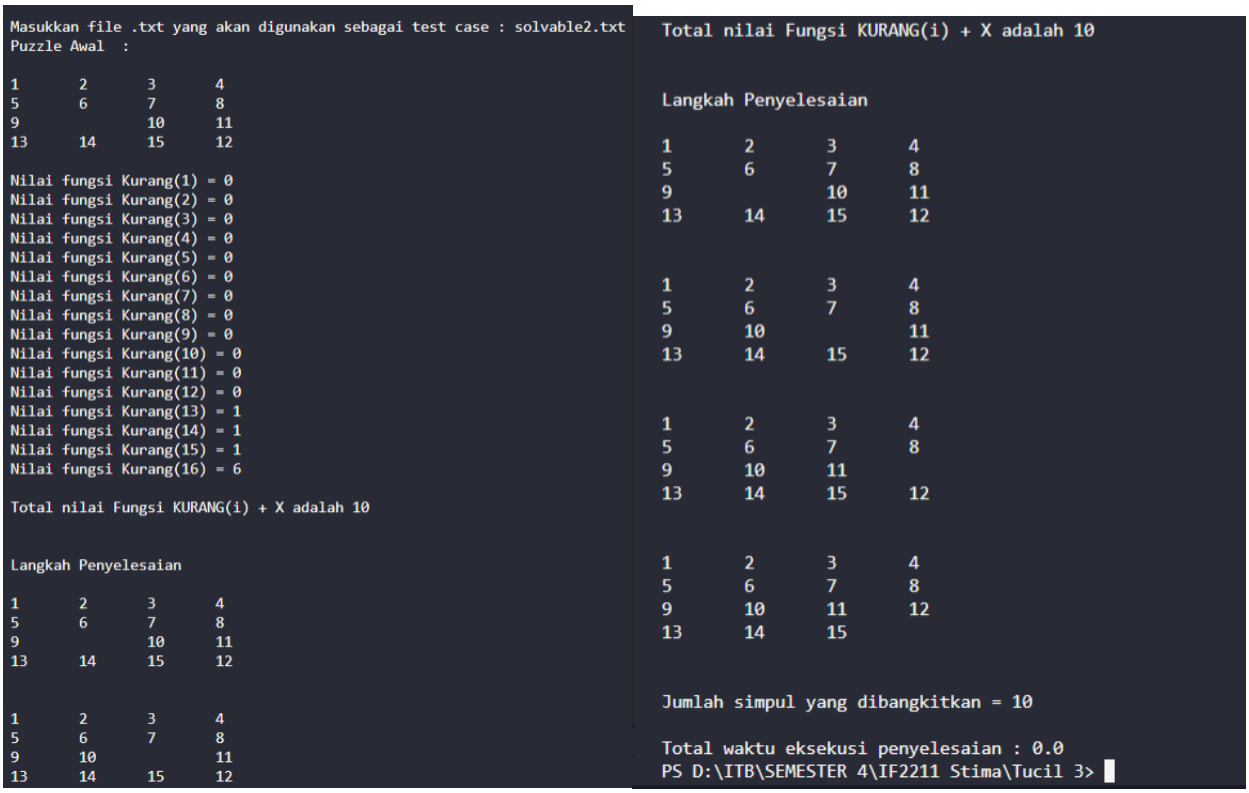




Input:



Output:



Input:

15_puzzle.py | solvable3.txt

```
solvable3.txt
1   1 2 3 4
2   5 6 16 12
3   9 10 8 7
4   13 14 11 15
```

```
Masukkan file .txt yang akan digunakan sebagai test case : solvable3.txt
Puzzle Awal  :

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

Output

```
Masukkan file .txt yang akan digunakan sebagai test case : solvable3.txt
Puzzle Awal  :

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

Nilai fungsi Kurang(1) = 0
Nilai fungsi Kurang(2) = 0
Nilai fungsi Kurang(3) = 0
Nilai fungsi Kurang(4) = 0
Nilai fungsi Kurang(5) = 0
Nilai fungsi Kurang(6) = 0
Nilai fungsi Kurang(7) = 0
Nilai fungsi Kurang(8) = 1
Nilai fungsi Kurang(9) = 2
Nilai fungsi Kurang(10) = 2
Nilai fungsi Kurang(11) = 0
Nilai fungsi Kurang(12) = 5
Nilai fungsi Kurang(13) = 1
Nilai fungsi Kurang(14) = 1
Nilai fungsi Kurang(15) = 0
Nilai fungsi Kurang(16) = 9

Total nilai Fungsi KURANG(i) + X adalah 22


Langkah Penyelesaian

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


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


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


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


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


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


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


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


Jumlah simpul yang dibangkitkan = 39

Total waktu eksekusi penyelesaian : 0.0040013790130615234
```

```
Langkah Penyelesaian

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


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


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


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


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


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


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

Input:

15_puzzle.py | no_solution1.txt

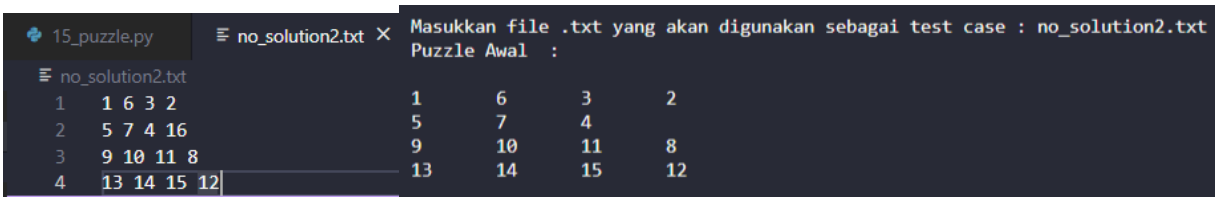
```
no_solution1.txt
1  1 3 4 15
2  2 16 5 12
3  7 6 11 14
4  8 9 10 13
```

```
Masukkan file .txt yang akan digunakan sebagai test case : no_solution1.txt
Puzzle Awal  :

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

Outptut:

```
Masukkan file .txt yang akan digunakan sebagai test case : no_solution1.txt
Puzzle Awal  :

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

Nilai fungsi Kurang(1) = 0
Nilai fungsi Kurang(2) = 0
Nilai fungsi Kurang(3) = 1
Nilai fungsi Kurang(4) = 1
Nilai fungsi Kurang(5) = 0
Nilai fungsi Kurang(6) = 0
Nilai fungsi Kurang(7) = 1
Nilai fungsi Kurang(8) = 0
Nilai fungsi Kurang(9) = 0
Nilai fungsi Kurang(10) = 0
Nilai fungsi Kurang(11) = 3
Nilai fungsi Kurang(12) = 6
Nilai fungsi Kurang(13) = 0
Nilai fungsi Kurang(14) = 4
Nilai fungsi Kurang(15) = 11
Nilai fungsi Kurang(16) = 10

Total nilai Fungsi KURANG(i) + X adalah 37


GA BISA DISELESAIIN NICH
```

Input:



output

```
Masukkan file .txt yang akan digunakan sebagai test case : no_solution2.txt
Puzzle Awal  :

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

Nilai fungsi Kurang(1) = 0
Nilai fungsi Kurang(2) = 0
Nilai fungsi Kurang(3) = 1
Nilai fungsi Kurang(4) = 0
Nilai fungsi Kurang(5) = 1
Nilai fungsi Kurang(6) = 4
Nilai fungsi Kurang(7) = 1
Nilai fungsi Kurang(8) = 0
Nilai fungsi Kurang(9) = 1
Nilai fungsi Kurang(10) = 1
Nilai fungsi Kurang(11) = 1
Nilai fungsi Kurang(12) = 0
Nilai fungsi Kurang(13) = 1
Nilai fungsi Kurang(14) = 1
Nilai fungsi Kurang(15) = 1
Nilai fungsi Kurang(16) = 8

Total nilai Fungsi KURANG(i) + X adalah 21


GA BISA DISELESAIIN NICH
```

Instansiasi Persoalan

Bisa diselesaikan :

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

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

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

Tidak bisa Diselesaikan :

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

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

Checklist

Poin	YA	TIDAK
1. Program Berhasil dikompilasi	V	
2. Program berhasil *running*	V	
3. Program dapat menerima input dan menuliskann ouput	V	
4. Luaran sudah benar untuk semua data uji	V	
5. Bonus dibuat		V

Link Program : https://github.com/Fikri-IF/STIMA-TUCIL-3