TUGAS KECIL III [IF2211] STRATEGI ALGORITMA
IMPLEMENTASI ALGORITMA BFS DAN A* PADA PERSOALAN LABIRIN (*MAZE PROBLEM*)

Dibuat oleh:

Leonardo / 13517048

Vinsen Marselino Andreas / 13517054

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2019

BAB I
KODE PROGRAM

Pada tugas besar kali ini kami menggunakan 5 buah kode sumber dengan bahasa pemrograman Python.

1. MainProg.py

```
# Tugas Kecil 3 IF2211 Strategi Algoritma
# NIM / Nama    : 13517048 / Leonardo
#                 13517054 / Vinsen Marselino Andreas
# Nama File     : MainProg.py
# Deskripsi     : Memanggil algoritma A* dan BFS untuk mendapatkan path pada maze

from colorama import init
from AStar import *
from BFS import *
from EksFile import *

if __name__ == '__main__':
    init() #colorama
    try:
        file_name = str(input("Masukkan nama file yang berisi matriks(.txt): "))
        matAStar = AmbilData(file_name)
        matBFS = AmbilData(file_name)
        print("Maze =")
        matAStar.Print()
        x0 = input("Masukkan x1 dan y1 (dipisah spasi) = ")
        x1, y1= int(x0.split()[0]), int(x0.split()[1])
        x0 = input("Masukkan x2 dan y2 (dipisah spasi) = ")
        x2, y2 = int(x0.split()[0]), int(x0.split()[1])
        start_point = Point(x1, y1)
        end_point = Point(x2, y2)
        print("\nHasil Algoritma A* =")
        try:
            path = Astar(matAStar, start_point, end_point)
            for ea in path:
                matAStar.AddData(ea.getX(), ea.getY(), 2) #berikan nilai 2 agar path
diwarnai hijau
            matAStar.Print()
        except (TypeError):
            print("Path tidak ditemukan!")

        print("\nHasil Algoritma BFS =")
        try:
            path = BFS(matBFS, start_point, end_point)
```

```
            for ea in path:
                matBFS.AddData(ea.getX(), ea.getY(), 2) #berikan nilai 2 agar path
diwarnai hijau
            matBFS.Print()
        except (TypeError):
            print("Path tidak ditemukan!")
    except (FileNotFoundError):
        print("File tidak ditemukan")
```

2. EksFile.py

```
# Tugas Kecil 3 IF2211 Strategi Algoritma
# NIM / Nama    : 13517048 / Leonardo
#                 13517054 / Vinsen Marselino Andreas
# Nama File     : EksFile.py
# Deskripsi     : Pembacaan file
from ADT import *

def AmbilData(file_name):
    list_input = []
    n_kolom = 0
    n_baris = 1
    check_baris = True

    mat_file = open(file_name,'r')
    a = mat_file.read(1)
    while (a!=""):
        if(a == '1' or a == '0'):
            list_input.append(a)
            if(check_baris):
                n_kolom += 1
        elif(a == '\n'):
            n_baris += 1
            check_baris = False
        a = mat_file.read(1) #baca 1 karakter

    iter_baris = 0
    iter_kolom = 0
    matriks_maze = Matriks(n_baris,n_kolom)
    for i in list_input:
        matriks_maze.AddData(iter_baris,iter_kolom,i)
        iter_kolom += 1
        if(iter_kolom == n_kolom):
            iter_baris += 1
```

```
            iter_kolom = iter_kolom % n_kolom

    return matriks_maze
```

3. ADT.py

```
# Tugas Kecil 3 IF2211 Strategi Algoritma
# NIM / Nama    : 13517048 / Leonardo
#                 13517054 / Vinsen Marselino Andreas
# Nama File     : ADT.py
# Deskripsi     : Tipe bentukan untuk Point, Matriks, dan Node

from colorama import Back, Style
import numpy as np
from math import sqrt

class Point:
    def __init__(self, _x = 0, _y = 0):
        self.x = _x
        self.y = _y

    def __eq__(self, other):
        return ((self.x == other.x) and (self.y == other.y))

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    @staticmethod
    def EuclideanDistance(P1, P2): #P1 dan P2 = Point
        return sqrt((P1.x - P2.x)**2 + (P1.y - P2.y)**2)

#matriks NxM
class Matriks:
    def __init__(self, _n = 0, _m = 0):
        self.N = _n
        self.M = _m
        self.data = np.zeros((self.N, self.M), dtype = int)

    def AddData(self, i, j, value):
        self.data[i][j] = value
```

```
    def getData(self, i, j):
        return self.data[i][j]

    def Print(self):
        for i in range(self.N):
            for j in range(self.M):
                if (self.getData(i,j) == 1):
                    print(Back.LIGHTBLACK_EX + " ", end = " ")
                    print(Style.RESET_ALL, end = "")
                elif (self.getData(i,j) == 2):
                    print(Back.GREEN + " ", end = " ")
                    print(Style.RESET_ALL, end = "")
                else: #self.getData(i,j) == 0
                    print(Back.WHITE + " ", end = " ")
                    print(Style.RESET_ALL, end = "")
            print(Style.RESET_ALL)

class Node:
    def __init__(self, _pred = None, _pos = Point()):
        self.pred = _pred
        self.pos = _pos

        self.toN = 0
        self.fromN = 0

    def isEqual(self, node2):
        return ((self.pos.x == node2.pos.x) and
                (self.pos.y == node2.pos.y))

    def ExistIn(self, listOfNode):
        for v in listOfNode:
            if (self.isEqual(v)):
                return True
        #sampai bagian ini jika self tidak exist di listOfNode
        return False

    def search(self, listOfNode):
        for v in listOfNode:
            if (self.isEqual(v)):
                return v.pred
        #sampai bagian ini jika self tidak exist di listOfNode
        return False

    def g(self):
```

```
        return self.toN

    def h(self):
        return self.fromN

    def f(self):
        return self.g() + self.h()
```

4. BFS.py

```
# Tugas Kecil 3 IF2211 Strategi Algoritma
# NIM / Nama    : 13517048 / Leonardo
#                 13517054 / Vinsen Marselino Andreas
# Nama File     : BFS.py
# Deskripsi     : Implementasi algoritma BFS
from ADT import *

def BFS(maze, start, end):
    if(maze.data[start.x][start.y] == 0 and maze.data[end.x][end.y] == 0):
        visited = []
        queue = []
        curr_node = Node(None, start)

        queue.append(curr_node)
        while (queue != [] and curr_node.pos != end):
            curr_node = queue.pop(0)
            prev_node = curr_node
            if(not(curr_node.ExistIn(visited))):
                visited.append(curr_node)
                if(curr_node.pos.x+1 < maze.N):
                    if (maze.data[curr_node.pos.x+1][curr_node.pos.y] == 0):
                        next_pos = Point(curr_node.pos.x+1,curr_node.pos.y)
                        next_node = Node(prev_node, next_pos)
                        queue.append(next_node)
                if(curr_node.pos.x-1 >= 0):
                    if (maze.data[curr_node.pos.x-1][curr_node.pos.y] == 0):
                        next_pos = Point(curr_node.pos.x-1,curr_node.pos.y)
                        next_node = Node(prev_node, next_pos)
                        queue.append(next_node)
                if(curr_node.pos.y+1 < maze.M):
                    if (maze.data[curr_node.pos.x][curr_node.pos.y+1] == 0):
                        next_pos = Point(curr_node.pos.x,curr_node.pos.y+1)
                        next_node = Node(prev_node, next_pos)
                        queue.append(next_node)
```

```
                if(curr_node.pos.y-1 >= 0):
                    if (maze.data[curr_node.pos.x][curr_node.pos.y-1] == 0):
                        next_pos = Point(curr_node.pos.x,curr_node.pos.y-1)
                        next_node = Node(prev_node, next_pos)
                        queue.append(next_node)
        if (curr_node.pos == end):
            path = [end]
            start_node = Node(None,end)
            while(start_node.pos != start):
                path_node = start_node.search(visited)
                path.append(path_node.pos)
                start_node = path_node
            path.append(start)
            return path[::-1]
```

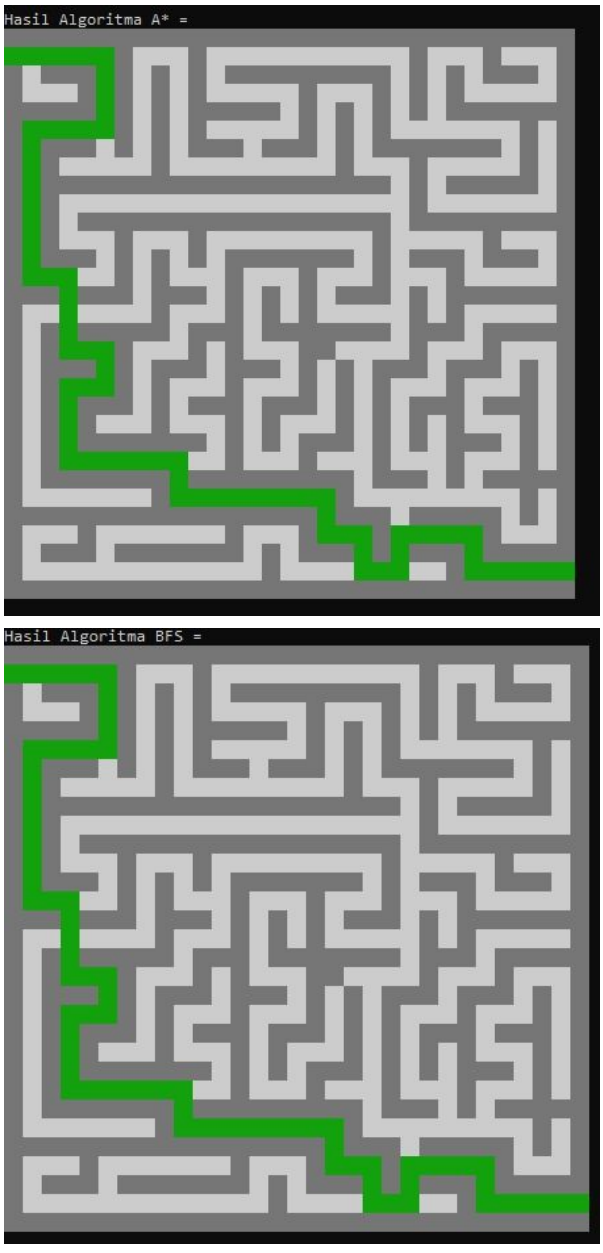
5. AStar.py

```
# Tugas Kecil 3 IF2211 Strategi Algoritma
# NIM / Nama    : 13517048 / Leonardo
#                 13517054 / Vinsen Marselino Andreas
# Nama File     : AStar.py
# Deskripsi     : Implementasi algoritma Astar

from ADT import *

#algoritma A Star
#matriks = Matriks
#Pstart, Pend = Point
def Astar(matriks, Pstart, Pend):
    if (matriks.getData(Pstart.getX(), Pstart.getY()) == 0): #dimulai dari jalan
        node_start = Node(None, Pstart)
        node_end = Node(None, Pend)

        candidate = []; visited = []
        candidate.append(node_start)
        #selagi masih ada kandidat
        while (len(candidate) > 0):
            idx = 0
            node_current = candidate[0]
            for i, n in enumerate(candidate):
                if (n.f() < node_current.f()):
                    idx = i
                    node_current = n #mengambil kandidat yang memiliki nilai f()
terkecil
```

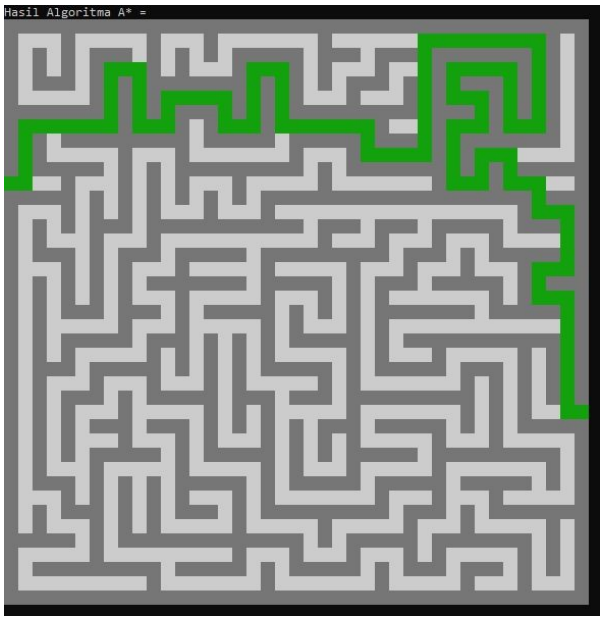
```
            candidate.pop(idx); visited.append(node_current)
            if (node_current.isEqual(node_end)):
                path = []
                current = node_current
                while current is not None:
                    path.append(current.pos)
                    current = current.pred
                return path[::-1] #dibalik

            else: #penelusuran belum selesai
                '''
                melakukan perjalanan sesuai arah jalan
                    0,1  = bergerak 1 ke atas
                    0,-1 = bergerak 1 ke bawah
                    1,0  = bergerak 1 ke kanan
                    -1,0 = bergerak 1 ke kiri
                '''
                arah_jalan = [[0, 1], [0, -1], [1, 0], [-1, 0]]
                successor = []
                for arah in arah_jalan:
                    temp_pos = Point(node_current.pos.getX() + arah[0],
                                     node_current.pos.getY() + arah[1])

                    if ((temp_pos.getX() < matriks.N) and
                        (temp_pos.getX() >= 0) and
                        (temp_pos.getY() < matriks.M) and
                        (temp_pos.getY() >= 0)):
                        #valid (ada di dalam maze)
                        if (matriks.getData(temp_pos.getX(), temp_pos.getY()) == 0):
                            #ada jalan
                            node_new = Node(node_current, temp_pos)
                            successor.append(node_new)

                for each in successor:
                    if (not each.ExistIn(visited)):
                        each.toN = node_current.g() + 1
                        each.fromN = Point.EuclideanDistance(each.pos, Pend)

                        if (not each.ExistIn(candidate)):
                            candidate.append(each)
    else: #Pstart tidak dimulai dari jalan yang ada
        raise TypeError
```

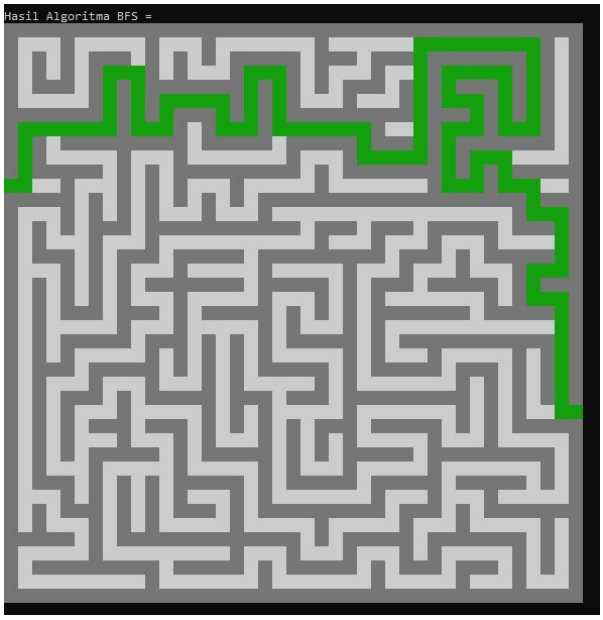
Teruntuk algoritma A*, g(n) kami merupakan berapa banyaknya langkah yang telah dilakukan, dan fungsi heuristik h(n) merupakan jarak *Euclidean* dari point sekarang menuju ke point finish.

BAB II
LABIRIN INPUT DAN SCREENSHOT HASIL

11111111111111111111 00001000100010010000 10101010111010101101 10100010000000101000 1 10111111111111110101 10000000001000001010 1 11111111101011101010 1 10001010001000100010 1 10101010111111111101 10100010001000000010 1 10111111101011110101 10100000101000001000 1 10101110101111101110 1 10001000001000101010 1 11111011101010101010 1 10001010101010101000 1 10101010101011101011 1 10101000101000001000 1 10111111101111111110 1 10000000000000000001 11111111111111111111	

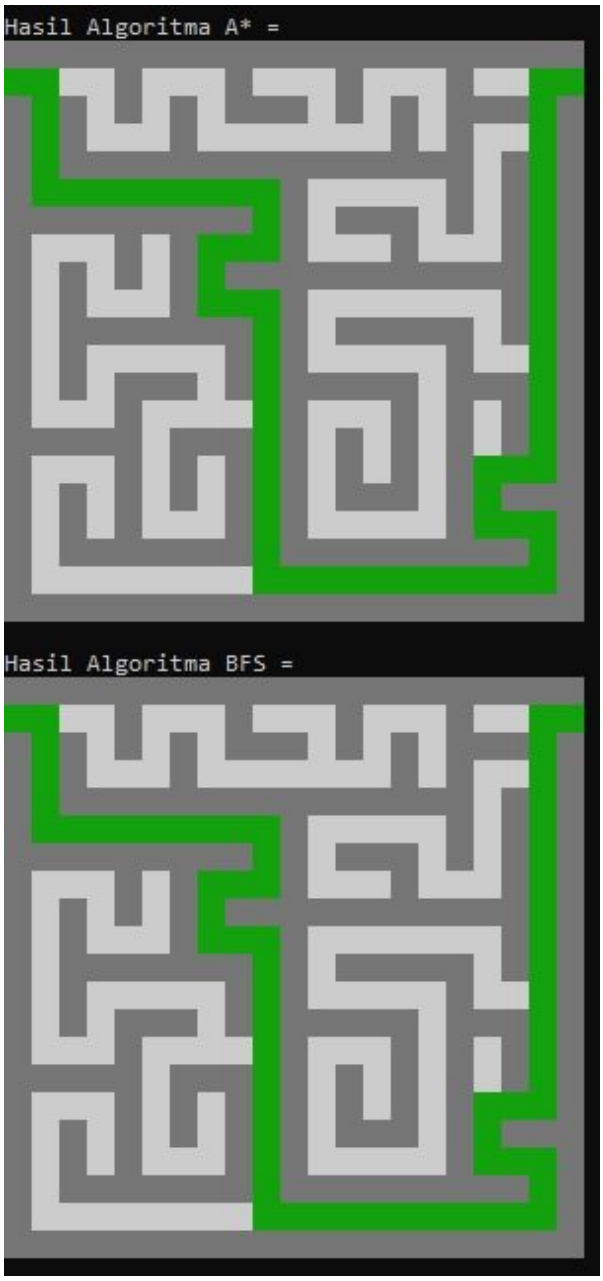
```
1111111111111111111111111111111111111111111111111
1000100000100010000000100000000000000000000000101
1010101110101010111110111011101111111110101010101
1010101000101000100010100010001000001010101010101
1011101010111111101010101110101011101010101010101
1000001010100000101010001000101000101010101010101
1111111010101110101011111111101110101010101010101
1000000010001010001000000010001000100010101010101
1010111111111011111011111011101011111110101010101
1010000010001000000010001000001010000000101010101
1011111010101111111110101111111010101111101010101
0000100010101000100000100000001000100000101010101
1111101010101010101111111111111111111110111010101
1000101010100010001000000000000000000010001010101
1010101110111111111111011101110111111011101010101
1010001000100000000000100010001000100000101010101
1011101011101111101111111111011101011111101010101
1000101010001000001000001000100010001000101010101
1010101010111111101111101011101111101011101010101
1010101010001000001000101010000000101000101010101
1010111011101011111010101011111110111110101010101
1010000010001000001010001010000000000000101010101
1010111110111010101011111010111111111111101010101
1010100000101010101000001010001000001010101010101
1011101111101010101111101011111011101010101010101
1010001000100010100000101000000010101010101010101
1010111010111111011101110111111111101010101010101
1010100010000010101000001000000010101000001010101
1010101110111010101010111011111010101111101010101
1010100010001010001010101000001000100000101010101
1011101010101111101011111011111011111010101010101
1000101010101000101000000010001000100000101010101
1010111010101110101111111111011101011111101010101
1010001010100000100000100000100010000010101010101
1111101011111111111111010111110111111101010101101
1000001000000000010001000100010100000101010101101
1011111111101111101011111010111011101010101010101
1000000000100000001000000010000010001000101010101
1111111111111111111111111111111111111111111111111
```





1111111111 1000001000 1001100011 1100111001 1000101111 1010101001 0010000011 1111111111	

```
11111111111111111111
00001000100010010000
10101010111010101101
10100010000000101001
10111111111111110101
10000000001000001010
11111111101011010101
10001010001000100101
10101010111111111101
10100010001000000101
10111111101011110101
10100000101000001001
10101110101111101101
10001000001000101010
11111011101010101010
10001010101010101001
10101010101011101011
10101000101000001001
10111111101111111101
10000000000000000001
11111111111111111111
```



111111111111 100000000001 001111110111 101000000001 101011111101 101010000001 100011011101 101000010111 101111000001 101011111111 100011000001 111111011111	Masukkan x1 dan y1 (dipisah spasi) = 2 0 Masukkan x2 dan y2 (dipisah spasi) = 10 10 Hasil Algoritma A* = Path tidak ditemukan! Hasil Algoritma BFS = Path tidak ditemukan!

BAB III
TABEL PENILAIAN

1.	Program dapat menerima input labirin	✔
2.	Program dapat mencari lintasan dengan Algoritma BFS	✔
3.	Program dapat mencari lintasan dengan Algoritma A*	✔
4.	Program dapat menampilkan lintasan di dalam labirin dengan algoritma BFS	✔
5.	Program dapat menampilkan lintasan di dalam labirin dengan algoritma a*	✔

BAB IV
DAFTAR PUSTAKA

1. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Tucil3-Stima-2019.pdf
2. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/BFS-dan-DFS-(2019).pdf
3. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf