Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

LAPORAN TUGAS KECIL 1

MATA KULIAH IF2211 STRATEGI ALGORITMA

Disusun oleh:

Juan Louis Rombetasik (13519075)

Aria Bachrul Ulum Berlian (13519115)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Semester 2 Tahun 2020/2021

Source Code Program

Modul read_input.py:

```
import os
from collections import defaultdict
from os.path import dirname, abspath

kamusKoordinat = defaultdict(dict)


def membaca_input():
    directory = dirname(abspath(__file__))
    namaFile = str(input("Nama File tanpa ekstensi: "))
    lokasiFile = os.path.join(directory, 'test\\' + namaFile + '.txt')
    f = open(lokasiFile, "r")

    kamusBeban = defaultdict(dict)

    # Mendapatkan list dari seluruh tempat, PENTING !
    listNodes = f.readline().replace("\n", "").split(" ")
    jumlahNodes = len(listNodes)

    # Parsing koordinat
    contents = f.readlines()
    arrayKoordinatMentah = contents[:contents.index("MATRIKS\n")]
    for koordinatTempat in arrayKoordinatMentah:
        namaNode = koordinatTempat.replace("\n", "").split(" ")
        kamusKoordinat[namaNode[0]]["lat"] = int(namaNode[1])
        kamusKoordinat[namaNode[0]]["lng"] = int(namaNode[2])

    # Parsing matriks
    MATRIKS = []
    arrayMatriksMentah = contents[contents.index("MATRIKS\n") + 1:]
    for elem in arrayMatriksMentah:
        elemenAdjacency = elem.replace("\n", "").split(" ")
        elemenAdjacency = list(map(int, elemenAdjacency))
        MATRIKS.append(elemenAdjacency)

    # Parsing bobot
    for ortu in range(jumlahNodes):
```

```
        for anak in range(jumlahNodes):
            if (MATRIKS[ortu][anak] == 1):
                kamusBeban[listNodes[ortu]][listNodes[anak]] =
jarakEuclidian(listNodes[ortu], listNodes[anak])
    return kamusBeban


def jarakEuclidian(start, end):
    return round(((kamusKoordinat[start]["lat"] -
kamusKoordinat[end]["lat"]) ** 2 + (
                kamusKoordinat[start]["lng"] - kamusKoordinat[end]["lng"])
** 2) ** (0.5), 3)
```

Modul visualisasiGraph:

```
import matplotlib.pyplot as pyplots
import networkx

from pencariJalur import Astar, kamusBeban
from read_input import kamusKoordinat


def membuatGraph(start, goal):
    # Inisialisasi Graph
    G = networkx.Graph()
    # Initialize variables from PathFinder
    ordered_sequence = Astar(start, goal)
    # Jarak
    distance_sum = 0
    for i in range(len(ordered_sequence) - 1):
        distance_sum += kamusBeban[ordered_sequence[i]][ordered_sequence[i
+ 1]]
    # Bentuk semua nodes
    for nodes in kamusBeban:
        G.add_node(nodes, pos=(kamusKoordinat[nodes]['lat'],
kamusKoordinat[nodes]['lng']))
    # Bentuk semua edges
    for nodes in kamusBeban:
        for children in kamusBeban[nodes]:
```

```
            G.add_edge(nodes, children,
weight=kamusBeban[nodes][children])

    # Posisi
    pos = networkx.get_node_attributes(G, 'pos')
    # Bobot
    labels = networkx.get_edge_attributes(G, 'weight')
    networkx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    # Mewarnai node yang dikunjungi
    node_color = []
    for node in G.nodes:
        if node in ordered_sequence:
            node_color.append("red")
        else:
            node_color.append("blue")

    networkx.draw(G, pos, with_labels=True, node_size=1200,
node_color=node_color)
    for i in range(len(ordered_sequence)):
        if (i != len(ordered_sequence) - 1):
            print(f"{ordered_sequence[i]} ->", end=" ")
        else:
            print(ordered_sequence[i])
    print(f"Panjang lintasan adalah {distance_sum}")
    pyplots.show()
```

Modul PencariJalur.py:

```
from read_input import membaca_input, jarakEuclidian

# total cost for nodes visited
kamusBeban = membaca_input()


def Astar(start, goal):
    kamusHeuristic = buatkamusHeuristic(goal)
    cost = {start: 0}

    # OPEN SET
    opened = []
```

```
    # CLOSE SET
    closed = []
    # CURRENT PLACE
    current = start
    # ADD CURRENT TO OPEN
    opened.append([current, kamusHeuristic[current]])
    while True:

        current = min(opened, key=lambda x: x[1])

        checked_node = current[0]

        closed.append(current)

        opened.remove(current)

        if (closed[-1][0] == goal):
            break

        for children in kamusBeban[checked_node].items():

            if children[0] in [closed_nodes[0] for closed_nodes in
closed]:
                continue

            cost.update({children[0]: cost[checked_node] + children[1]})

            current_fval = cost[checked_node] +
kamusHeuristic[children[0]] + children[1]

            temp = [children[0], current_fval]
            opened.append(temp)


    last_node = goal
    ordered_sequence = []
    ordered_sequence.append(goal)

    for i in range(len(closed) - 2, -1, -1):
```

```
        check_node = closed[i][0]

        if last_node in [children[0] for children in
kamusBeban[check_node].items()]:
            if (cost[check_node] + kamusBeban[check_node][last_node] ==
cost[last_node]):
                ordered_sequence.append(check_node)
                last_node = check_node
    # Reverse ordering from ordered_sequence
    ordered_sequence.reverse()
    return ordered_sequence


def buatkamusHeuristic(goal):
    kamusHeuristic = dict()
    for nodes in kamusBeban:
        kamusHeuristic[nodes] = jarakEuclidian(nodes, goal)
    return kamusHeuristic
```

Main.py:

```
from pencariJalur import kamusBeban
from visualiasiGraph import membuatGraph

while True:
    kamus = kamusBeban
    arrayNodes = [nodes for nodes in kamus]
    # List nama tempat
    print("Tempat-Tempat: ")
    num = 1
    for i in arrayNodes:
        print(f"{num}. {i}")
        num += 1
    # Input tempat awal
    tempatAwal = str(input("Tempat Awal: "))
    while (tempatAwal not in arrayNodes):
        print("Tempat awal tidak ditemukan")
        tempatAwal = str(input("Tempat Awal: "))
    # Input tempat akhir
    tempatAkhir = str(input("Tempat Akhir: "))
```

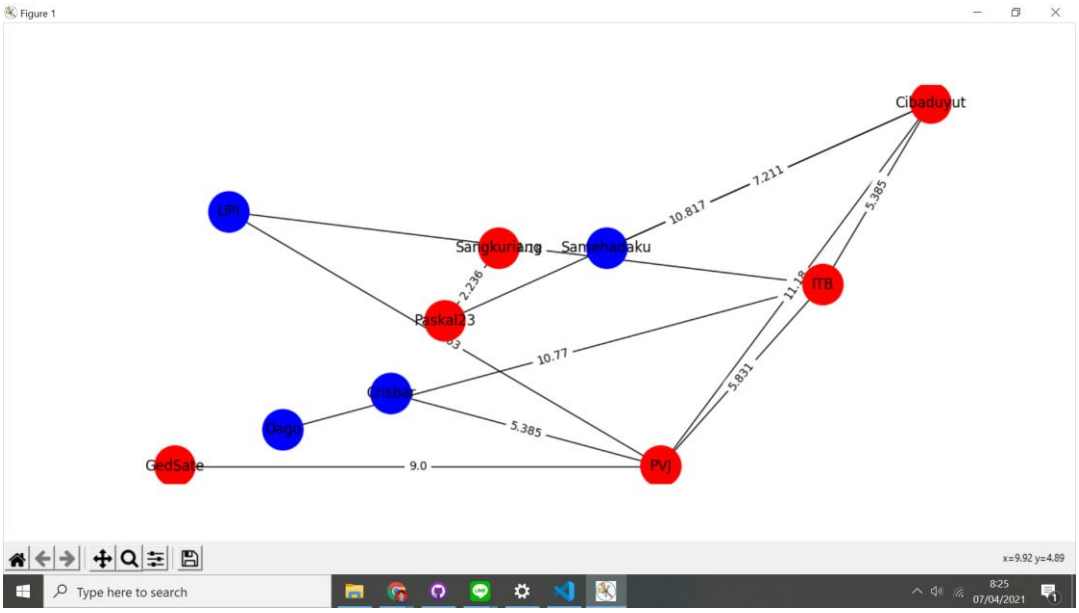
```
    while (tempatAkhir not in arrayNodes):
        print("Tempat akhir tidak ditemukan")
        tempatAkhir = str(input("tempat Akhir: "))
    # Membentuk graph
    membuatGraph(tempatAwal, tempatAkhir)
    break
```

Pengujian

1. **Input:**

   Crisbar Sangkuriang Dago Paskal23 GedSate PVJ UPI ITB Cibaduyut Samehadaku
   Crisbar 2 5
   Sangkuriang 4 9
   Dago 0 4
   Paskal23 3 7
   GedSate -2 3
   PVJ 7 3
   UPI -1 10
   ITB 10 8
   Cibaduyut 12 13
   Samehadaku 6 9
   MATRIKS
   0 0 0 0 0 1 0 0 0 0
   0 0 0 1 0 0 0 0 0 0
   0 0 0 0 0 0 0 1 0 0
   0 1 0 0 0 0 0 0 1 0
   0 0 0 0 0 1 0 0 0 0
   1 0 0 0 1 0 1 1 1 0
   0 0 0 0 0 1 0 1 0 0
   0 0 1 0 0 1 1 0 1 0
   0 0 0 1 0 1 0 1 0 1
   0 0 0 0 0 0 0 0 1 0

   Output:



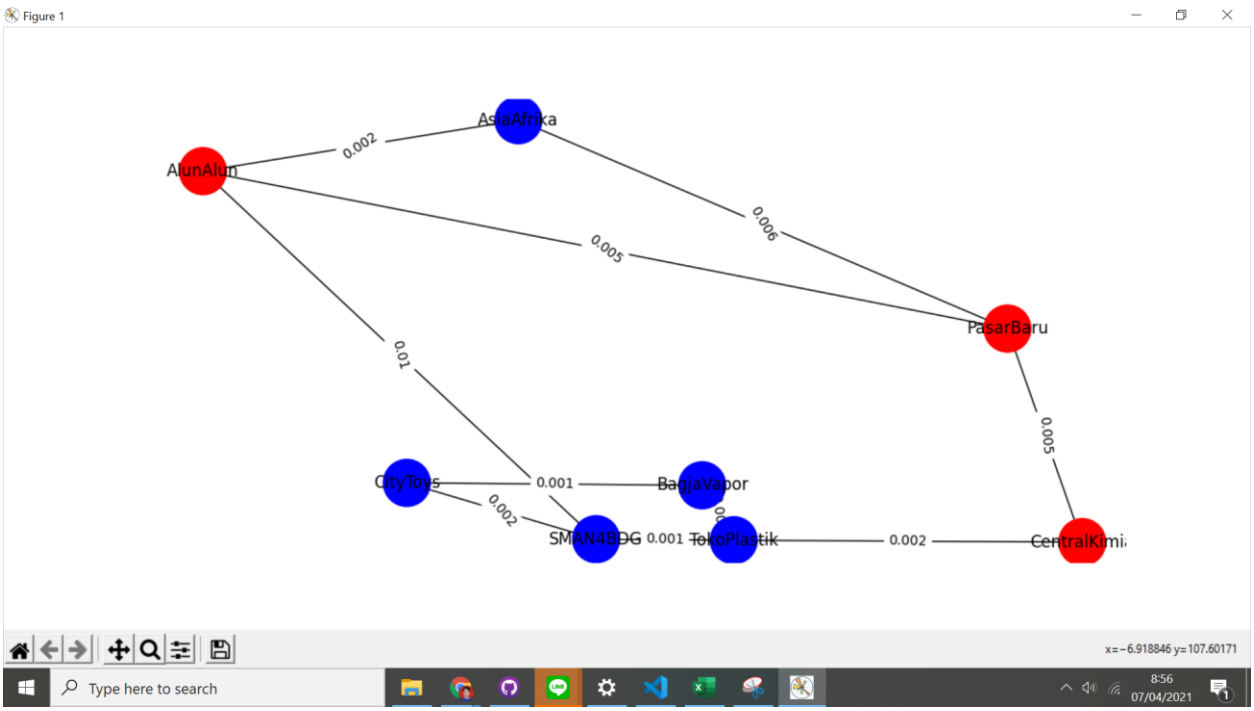
```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Kuliahan\SMT 4\Strategi Algoritma\Tucil\3\Tucil3_13519075>python -u Main.py
Nama File tanpa ekstensi: test1
Tempat-Tempat:
1. Crisbar
2. Sangkuriang
3. Dago
4. Paskal23
5. GedSate
6. PVJ
7. UPI
8. ITB
9. Cibaduyut
10. Samehadaku
Tempat Awal: Sangkuriang
Tempat Akhir: GedSate
Sangkuriang -> Paskal23 -> Cibaduyut -> ITB -> PVJ -> GedSate
Panjang lintasan adalah 33.269000000000005
```

2. **Input:**

SMAN4BDG TokoPlastik BagjaVapor CityToys CentralKimia PasarBaru AsiaAfrika AlunAlun
SMAN4BDG -6.919355872854605 107.5983220466102
TokoPlastik -6.918710922292898 107.59829497652909
BagjaVapor -6.918858723541105 107.59968908570696
CityToys -6.920242678438361 107.5997499933895
CentralKimia -6.917078387214849 107.59824760386712
PasarBaru -6.917428835551521 107.603663930168
AsiaAfrika -6.919719755962046 107.60894936363049
AlunAlun -6.921197763221031 107.60765676724947
MATRIKS
0 1 0 1 0 0 0 1
1 0 1 0 1 0 0 0
0 1 0 1 0 0 0 0
1 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0
0 0 0 0 1 0 1 1
0 0 0 0 0 1 0 1
1 0 0 0 0 1 1 0

Output:



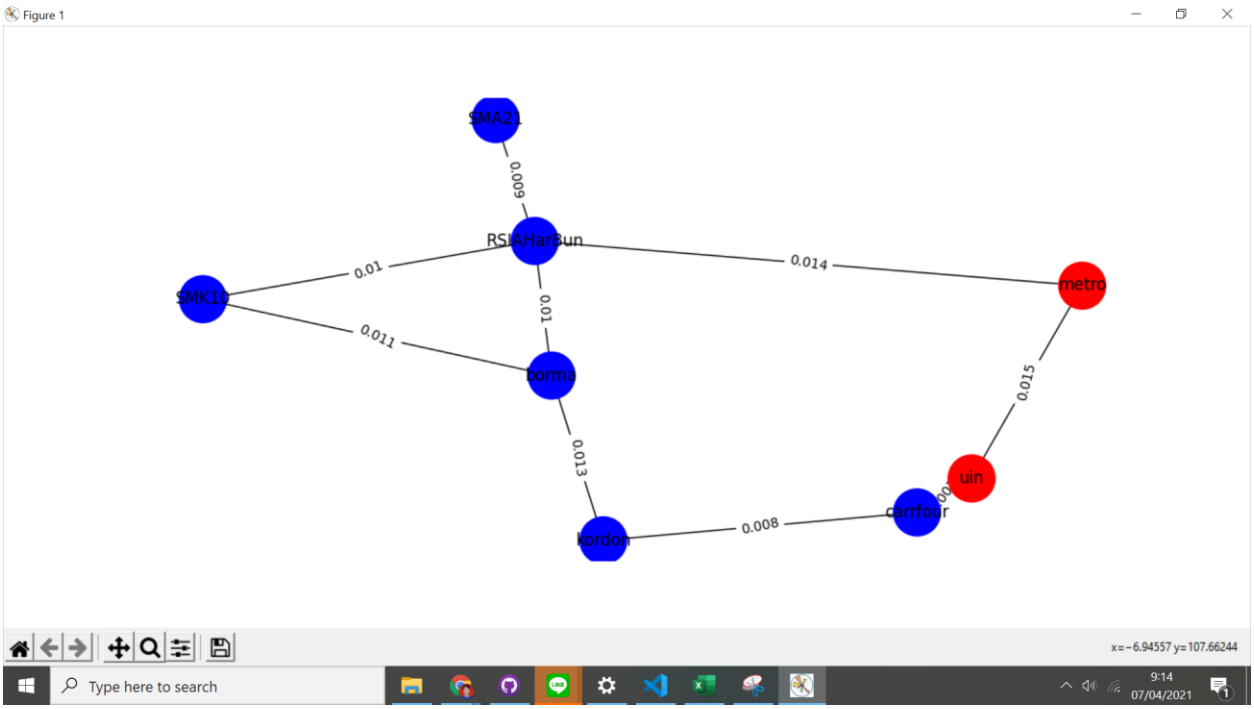

```
D:\Kuliahan\SMT 4\Strategi Algoritma\Tucil\3\Tucil3_13519075>python Main.py
Nama File tanpa ekstensi: test2
Tempat-Tempat:
1. SMAN4BDG
2. TokoPlastik
3. BagjaVapor
4. CityToys
5. CentralKimia
6. PasarBaru
7. AsiaAfrika
8. AlunAlun
Tempat Awal: CentralKimia
Tempat Akhir: AlunAlun
CentralKimia -> PasarBaru -> AlunAlun
Panjang lintasan adalah 0.01
```

3. **Input:**

carrfour uin metro RSIAHarBun SMA21 borma SMK10 kordon
carrfour -6.945830761014347 107.64116338702699
uin -6.944436099652653 107.64379320354575
metro -6.941611003599316 107.65870750560326
RSIAHarBun -6.955593274622046 107.6621658945049
SMA21 -6.95659454670303 107.67160441417965
borma -6.9551641575602385 107.65175470305766
SMK10 -6.964068259479718 107.65766278400392
kordon -6.953841043588939 107.63900189390823
MATRIKS
0 1 0 0 0 0 0 1
1 0 1 0 0 0 0 0

0 1 0 1 0 0 0 0
0 0 1 0 1 1 1 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 1 1
0 0 0 1 0 1 0 0
1 0 0 0 0 1 0 0

Output:

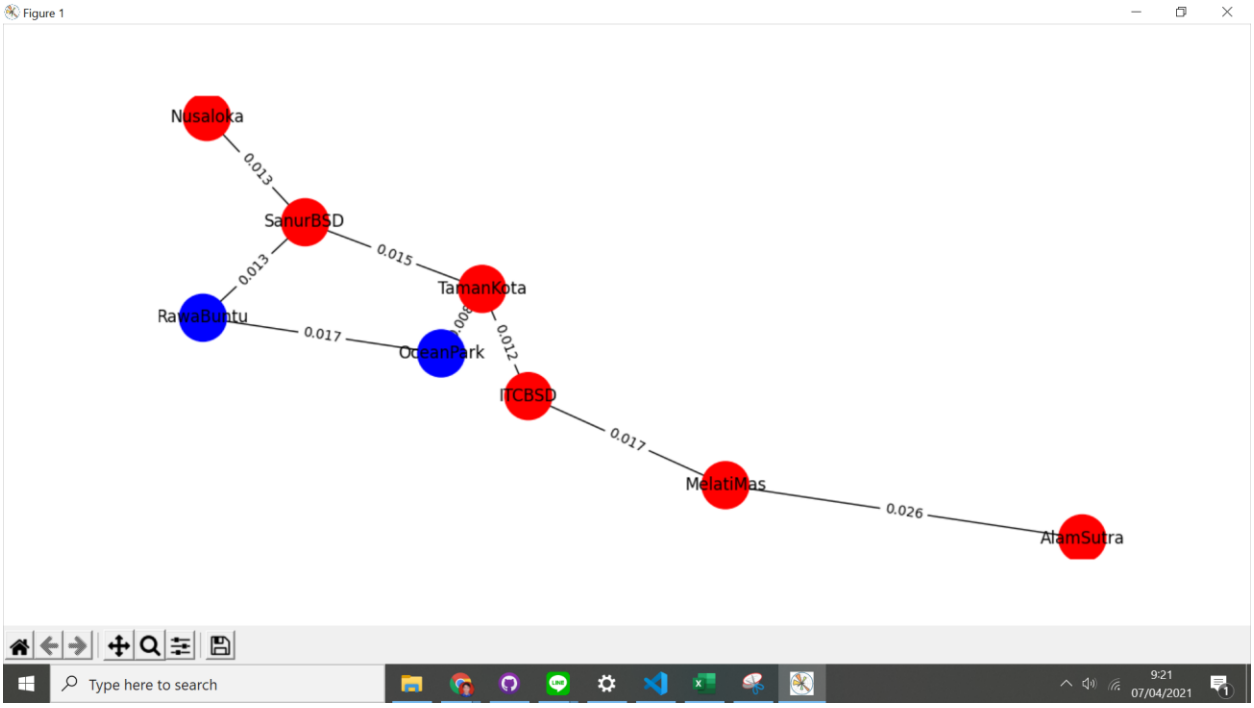



```
D:\Kuliahan\SMT 4\Strategi Algoritma\Tucil\3\Tucil3_13519075>python Main.py
Nama File tanpa ekstensi: test3
Tempat-Tempat:
1. carrfour
2. uin
3. metro
4. RSIAHarBun
5. SMA21
6. borma
7. SMK10
8. kordon
Tempat Awal: uin
Tempat Akhir: metro
uin -> metro
Panjang lintasan adalah 0.015
```

4. **Input:**
   Nusaloka SanurBSD RawaBuntu OceanPark TamanKota ITCBSD MelatiMas AlamSutra
   Nusaloka -6.3078931813917825 106.69481904600853
   SanurBSD -6.300862867264631 106.68350670893508
   RawaBuntu -6.3081739031640005 106.6732303351251

OceanPark -6.291125320280125 106.6694002731189
TamanKota -6.288193456680016 106.67633195433625
ITCBSD -6.284888953878086 106.66480208683579
MelatiMas -6.270779661108461 106.65522554995633
AlamSutra -6.2452464900434785 106.64950717634011
MATRIKS
0 1 0 0 0 0 0 0
1 0 1 0 1 0 0 0
0 1 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 1 0 1 0 1 0 0
0 0 0 0 1 0 1 0
0 0 0 0 0 1 0 1
0 0 0 0 0 0 1 0

Output:



```
D:\Kuliahan\SMT 4\Strategi Algoritma\Tucil\3\Tucil3_13519075>python Main.py
Nama File tanpa ekstensi: test4
Tempat-Tempat:
1. Nusaloka
2. SanurBSD
3. RawaBuntu
4. OceanPark
5. TamanKota
6. ITCBSD
7. MelatiMas
8. AlamSutra
Tempat Awal: Nusaloka
Tempat Akhir: AlamSutra
Nusaloka -> SanurBSD -> TamanKota -> ITCBSD -> MelatiMas -> AlamSutra
Panjang lintasan adalah 0.08299999999999999
```

		Centang (√) jika ya
1	Program dapat menerima input graf	√
2	Program dapat menghitung lintasan terpendek	√
3	Program dapat menampilkan lintasan terpendek serta jaraknya	√
4	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	

Source Code : https://github.com/ariaberlian/Tucil3_13519075