LAPORAN TUGAS KECIL 3
IF2211 STRATEGI ALGORITMA
Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Oleh :

Mahameru Ds	**13519014**
Ilyasa Salafi Putra Jamal	**13519023**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG BANDUNG

2021

Kode Program

1. File PriorityQueue.py

```
# Kelas isi/item priority queue
class PriorityQueueItem :
    def __init__(self, item, priority) :
        if(type(priority) != str and type(priority) != int and type(
priority) != float) :
            raise Exception("QueueItem priority must be a string or
int or float")
        self.item = item
        self.priority = priority
    def __str__(self) :
        return "(" + str(self.item) + "," + str(self.priority) + ")"

# Kelas priority queue
class PriorityQueue :
    def __init__(self) :
        self.values = []
        self.length = 0
    def __str__(self) :
        text = "["
        for i in range(self.length):
            item = self.values[i]
            text += str(item)
            if(i != self.length-1) :
                text += ","
        text += "]"
        return text

    def enqueue(self, item, priority) :
        newItem = PriorityQueueItem(item, priority)
        self.values.append(newItem)
        self.length += 1
        self.sort()

    def dequeue(self) :
        item = self.values.pop(0)
        self.length -= 1
        return item

    # Sort berdasarkan priority
    def sort(self) :
        for i in range(self.length):
```

```
            for j in range(self.length):
                item1 = self.values[i]
                item2 = self.values[j]
                if(item1.priority < item2.priority) :
                    temp = item1
                    self.values[i] = item2
                    self.values[j] = temp

    # Cek apakah ada item
    def exists(self, item) :
        for val in self.values :
            if(val.item == item) :
                return True
        return False

    # Cek apakah kosong
    def empty(self) :
        return self.length == 0
```

2. File Node.py

```
# Kelas simpul
class Node :
    def __init__(self, title, x, y) :
        if(type(title) != str ) :
            raise Exception("Node title argument must be a string")
        if(type(x) != int or type(y) != int) :
            raise Exception("X and Y argument must be an integer")
        self.title = title
        self.x = x
        self.y = y
    def __str__(self) :
        return "(" + self.title + "," + str(self.x) + "," + str(self
.y) + ")"
```

3. File Graph.py

```
import copy

from Node import Node
from PriorityQueue import PriorityQueue

# Directed Graph
```

```
class Graph :
    def __init__(self) :
        # Node Coordinates
        self.__nodes = {}

        # Edges Weight/Priority
        self.__adjacencyList = {}

    # Menampilkan semua node
    def printNodes(self) :
        print("[ ", end="")
        for node in self.nodes :
            print(node, end=" ")
        print(" ]")

    # Menambahkan simpul baru
    def addNode(self, nodeTitle, x, y) :
        newNode = Node(nodeTitle, x, y)
        self.__nodes[nodeTitle] = (newNode)
        self.__adjacencyList[newNode.title] = {}

    # Menambahkan sisi baru
    def addEdge(self, nodeStart, nodeEnd, weight) :
        # Error Handling
        if(type(nodeStart) != str or type(nodeEnd) != str ) :
            raise Exception("Node argument must be a string")
        if(type(weight) != int) :
            if(weight != float('inf')) :
                raise Exception("Weight argument must be an integer
or infinity")
        self.__adjacencyList[nodeStart][nodeEnd] = weight

    # Mendapatkan jarak Eucledean
    def getEuclideanDistance(self, x1, x2, y1, y2) :
        return ((x1 - x2)**2 + (y1 - y2)**2)**(0.5)

    # Heuristik
    def getHeuristic(self, nodeStart, nodeEnd) :
        if(type(nodeStart) != Node or  type(nodeEnd) != Node) :
            raise Exception("Argument must be an instance of Node Ob
ject")
        return self.getEuclideanDistance(nodeStart.x, nodeEnd.x, nod
eStart.y, nodeEnd.y);

    def shortestPath(self, nodeStart, nodeEnd) :
```

```
        if(type(nodeStart) != str or type(nodeEnd) != str) :
            raise Exception("Shortest path node argument must be a s
tring")
        # Shortest Path Using A* Algorithm
        nodeSet = PriorityQueue()
        distances = {}
        previous = {}
        visited = []
        # Initial State
        nodeSet.enqueue(nodeStart, 0)
        previous[nodeStart] = None
        # Filling distances with inf
        for node in self.__nodes :
            if(node != nodeStart) :
                distances[node] = float('inf')
            else :
                distances[nodeStart] = 0
        while not nodeSet.empty():
            currentNode = nodeSet.dequeue().item;
            if(currentNode == nodeEnd) :
                path = []
                # distances = 0
                while(previous[currentNode] != None) :
                    path.insert(0,currentNode);
                    currentNode = previous[currentNode];
                path.insert(0, nodeStart)
                print(distances)
                return path
            adjacencyList = dict(filter(lambda node: node[1] != floa
t('inf'), self.__adjacencyList[currentNode].items()))
            for neighbor in adjacencyList :
                if(neighbor in visited) :
                    continue
                weight = self.__adjacencyList[currentNode][neighbor]

                if(weight == float('inf')) :
                    continue
                candidateDistance = distances[currentNode] + int(wei
ght)
                if(candidateDistance < distances[neighbor]) :
                    previous[neighbor] = currentNode
                    distances[neighbor] = candidateDistance
                    nodeStartInput = self.__nodes[nodeStart]
                    neighborInput = self.__nodes[neighbor]
```

```
                    priority = distances[neighbor] + self.getHeurist
ic(nodeStartInput, neighborInput)
                    if not nodeSet.exists(neighbor) :
                        nodeSet.enqueue(neighbor, priority)
            visited.append(currentNode)

    # Membuat graf dari file
    def setGraphFromFile(self, filename) :
        # Read the file
        f = open(filename, "r", encoding="utf-8")
        arr = f.readlines()
        temp = []
        for i in arr :
            temp.append(i.rstrip())
        arr = temp
        nodeCount = int(arr[0])
        nodes = arr[1:nodeCount+1]
        adjacencyList = arr[nodeCount+1:]
        for node in nodes :
            nodeAttributes = node.split();
            title = nodeAttributes[0]
            x = int(nodeAttributes[1])
            y = int(nodeAttributes[2])
            self.addNode(title, x, y);
        for connection in adjacencyList :
            connectionList = connection.split();
            nodeStart = connectionList[0]
            for i in range(len(connectionList[1:])) :
                key = list(self.__nodes.keys())[i]
                nodeEnd = self.__nodes[key].title
                weight = int(connectionList[1:][i])
                if(weight == -1) :
                    weight = float('inf')
                self.addEdge(nodeStart, nodeEnd, weight)
```

4. File utils.py

```
import random
import copy

from Graph import Graph

# Cek graf siklik/tidak
```

```
def checkAcyclic(node, graph, visited, origin) :
    # Get the key
    key = node if type(node) == str else node.title
    connection = graph.connections[key]

    # Add the current node to the visited node
    visited.append(key)
    for node in connection :
        if(node in visited) :
            return False
        else :
            # If node has not been visited, check the node.
            return checkAcyclic(node, graph, visited, origin)

    # If all nodes has been visited return true
    connection = graph.connections[origin.title]
    count = 0
    for node in connection :
        if(node in visited) :
            count += 1
    return count == len(connection)

# Buat graf asiklik
def generateAcyclicGraph(numberOfNodes, seed) :
    graph = Graph()
    for i in range(numberOfNodes) :
        graph.addNode(f"C{i+1}")
    for i in range(seed) :
        # Select a random node
        number1 = random.randint(0, numberOfNodes-1)
        number2 = random.randint(0, numberOfNodes-1)
        while number1 == number2 :
            number1 = random.randint(0, numberOfNodes-1)
            number2 = random.randint(0, numberOfNodes-1)

        # Create a temporary graph
        tempGraph = copy.deepcopy(graph)
        node1 = tempGraph.nodes[number1]
        node2 = tempGraph.nodes[number2]
        connection = tempGraph.connections[node1.title]

        # Check if the current node has been connected before
        if(node2.title not in connection) :
            tempGraph.addConnection(node1,node2)
```

```
            # Check if by adding the new connection, the graph has b
ecome cyclic
            if(checkAcyclic(node1, tempGraph, [], node1)) :
                # Add the connection to the actual graph
                graph.addConnection(node1,node2)
    return graph

def generateTestCases(numOfCases, nodes) :
    j = 0
    for i in range(numOfCases) :
        graph = generateAcyclicGraph(nodes, 1000)
        result = graph.topologicalSort()
        # Pembuatan test case terbatas hanya pada test case yang mem
iliki setidaknya depth=8
        if(len(result) == 8) :
            graph.writeGraphToFile(f"./testCase/{j+1}.txt")
            j+= 1

def resultToText(result, filename) :
    f = open(filename, "w", encoding="utf-8")
    string = ""
    finishedFile = ""
    i = 0
    for key in result :
        string += "Semester " + str(i+1) + " : {"
        for node in result[key] :
            string += " " + node + " "
        string += "}\n"
        finishedFile += string
        string = ""
        i += 1
    f.write(finishedFile)
    f.close()
```

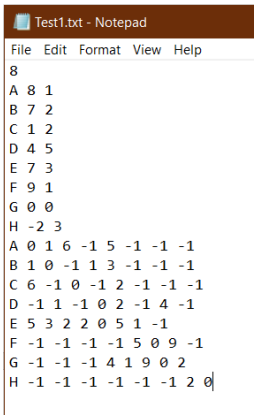
5. File main.py

```
from Node import Node
from Graph import Graph
from utils import generateTestCases, resultToText

import os
# Dengan mengasumsikan tiap case diberi nama Test1.txt, Test2.txt, d
st
```

```
def runTestCases(numOfCases) :
    for i in range(numOfCases) :
        filename = f"Test{i+1}.txt"
        graph = Graph()
        graph.setGraphFromFile("Test_Case/" + filename)
        start = input("Masukkan simpul asal: \n")
        end = input("Masukkan simpul tujuan: \n")
        path = graph.shortestPath(start, end)
        print(path)

def main() :
    # Tinggal memasukkan nama file beserta ekstensinya, Contoh: "Tes
t1.txt"
    filename = input("Masukkan nama file: \n")
    graph = Graph()
    graph.setGraphFromFile("../test/" + filename)

    # Pastikan nama simpul sesuai dengan pada file input
    start = input("Masukkan simpul asal: \n")
    end = input("Masukkan simpul tujuan: \n")
    path = graph.shortestPath(start, end)
    print(path)

main()
```

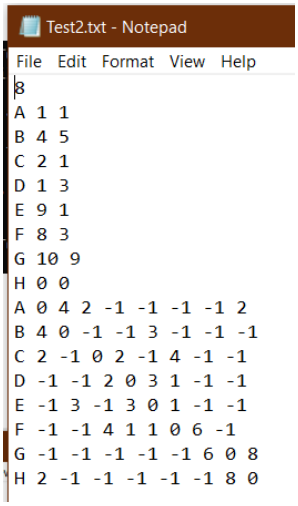
Hasil Uji

Tes 1:




```
C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test1.txt
Masukkan simpul asal:
A
Masukkan simpul tujuan:
H
{'A': 0, 'B': 1, 'C': 6, 'D': 2, 'E': 4, 'F': 9, 'G': 5, 'H': 7}
['A', 'B', 'E', 'G', 'H']

C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>
```

Tes 2:




```
C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test2.txt
Masukkan simpul asal:
A
Masukkan simpul tujuan:
G
{'A': 0, 'B': 4, 'C': 2, 'D': 4, 'E': 6, 'F': 5, 'G': 10, 'H': 2}
['A', 'H', 'G']
```

Tes 3:

Test3.txt - Notepad

File Edit Format View Help

```
8
kubus -933 104
bonbin -936 84
batan -882 80
sabuga -863 78
dayang -874 115
saraga -856 102
mcdonalds -852 135
borromeus -937 130
kubus 0 27 30 -1 -1 -1 -1 -1
bonbin 27 0 60 -1 -1 -1 -1 -1
batan -1 60 0 62 43 -1 -1 -1
sabuga -1 -1 62 0 54 -1 -1 -1
dayang -1 -1 43 54 0 83 55 -1
saraga -1 -1 -1 -1 83 0 74 -1
mcdonalds -1 -1 -1 -1 55 74 0 98
borromeus 30 -1 -1 -1 -1 -1 98 0
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test3.txt
Masukkan simpul asal:
kubus
Masukkan simpul tujuan:
borromeus
{'kubus': 0, 'bonbin': 27, 'batan': 30, 'sabuga': 92, 'dayang': 73, 'saraga': 156, 'mcdonalds': 128, 'borromeus': 226}
['kubus', 'batan', 'dayang', 'mcdonalds', 'borromeus']

C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>
```

Tes 4:

Test4.txt - Notepad

File Edit Format View Help

```
8
masjid -218 682
parahyangan -224 629
pendopo -233 701
pln -213 835
museum -211 985
fountain -200 896
sma -239 751
gereja -251 606
masjid 0 81 154 244 -1 -1 407 -1
parahyangan 81 0 152 -1 -1 -1 382 307
pendopo 154 152 0 326 -1 -1 357 -1
pln 407 -1 326 0 214 175 446 -1
museum -1 -1 -1 214 0 277 -1 -1
fountain -1 -1 -1 175 277 0 -1 -1
sma 407 382 357 446 -1 -1 0 267
gereja -1 307 -1 -1 -1 -1 267 0
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test4.txt
Masukkan simpul asal:
masjid
Masukkan simpul tujuan:
museum
{'masjid': 0, 'parahyangan': 81, 'pendopo': 154, 'pln': 244, 'museum': 458, 'fountain': 419, 'sma': 407, 'gereja': 388}
['masjid', 'pln', 'museum']

C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>
```

Tes 5:

Test5.txt - Notepad

File Edit Format View Help

```
8
Borma -554 519
SMPN18 -518 416
Carrefour -466 416
Uninus -645 444
Edelweiss -433 496
Metro -419 588
Dinas -343 625
Pandanwangi -627 515
Borma 0 16 -1 -1 22 -1 -1 13
SMPN18 16 0 6 -1 -1 -1 -1 -1
Carrefour -1 6 0 4 -1 -1 -1 -1
Uninus -1 -1 4 0 7 -1 -1 -1
Edelweiss 22 -1 -1 7 0 12 -1 -1
Metro -1 -1 -1 -1 12 0 15 -1
Dinas -1 -1 -1 -1 -1 15 0 -1
Pandanwangi 13 -1 -1 -1 -1 -1 -1 0
```

```
C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test5.txt
Masukkan simpul asal:
Borma
Masukkan simpul tujuan:
Dinas
{'Borma': 0, 'SMPN18': 16, 'Carrefour': 22, 'Uninus': 29, 'Edelweiss': 22, 'Metro': 34, 'Dinas': 49, 'Pandanwangi': 13}
['Borma', 'Edelweiss', 'Metro', 'Dinas']
```

Tes 6:

Test6.txt - Notepad

File Edit Format View Help

```
8
Revo -553 699
MM -48 699
Stadium -384 699
Summarecon -269 700
RSUD -422 700
Jakapermai -477 697
GGP -705 697
Kemang -698 699
Revo 0 91 -1 -1 -1 -1 -1 205
MM 91 0 112 -1 214 187 -1 -1
Stadium -1 112 0 134 190 -1 -1 -1
Summarecon -1 -1 134 0 -1 -1 -1 -1
RSUD -1 214 190 -1 0 -1 -1 -1
Jakapermai -1 187 -1 -1 -1 0 36 -1
GGP -1 -1 -1 -1 -1 36 0 25
Kemang 205 -1 -1 -1 -1 -1 25 0
```

```
C:\Tugas\Stima\Tucil3\Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm>py main.py
Masukkan nama file:
Test6.txt
Masukkan simpul asal:
Revo
Masukkan simpul tujuan:
Stadium
{'Revo': 0, 'MM': 91, 'Stadium': 203, 'Summarecon': inf, 'RSUD': 305, 'Jakapermai': 266, 'GGP': 230, 'Kemang': 205}
['Revo', 'MM', 'Stadium']
```

Alamat Source Code

https://github.com/eruds/Tugas-Kecil-3-Strategi-Algoritma-A-Algorithm

(Masih private)

Folder "../Tucil_13519014/src"

Spesifikasi Program

No.	Poin	Ya	Tidak
1.	Program dapat menerima input graf.	V	
2.	Program dapat menghitung lintasan terpendek.	V	
3.	Program dapat menampilkan lintasan terpendek serta jaraknya.	V	
4.	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta.		V