**Tugas Kecil 3 IF2122 Strategi Algoritma**

**Implementasi Algoritma A\* untuk Menentukan Lintasan Terpendek**

**LAPORAN**

Oleh

**DAFI IHSANDIYA FARAZ 13516057**

**RABBI FIJAR MAYOZA 13516081**

****

**TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2018**

**BAB I**

**KODE PROGRAM**

Program berikut dibuat menggunakan bahasa Python 2.7 dan *library* Matplotlib.

import random

import matplotlib.pyplot as plt

import time

# Tucil 2 Stima: Membentuk convex hull menggunakan algoritma QuickHull

# NIM : 13516081

# Fungsi generate points secara acak

def generatePoints(n):

radius = 10

rangeX = (0, 200)

rangeY = (0, 200)

qty = n # jumlah titik

deltas = set()

for x in range(-radius, radius+1):

for y in range(-radius, radius+1):

if x\*x + y\*y <= radius\*radius:

deltas.add((x,y))

randPoints = []

excluded = set()

i = 0

while i<qty:

x = random.randrange(\*rangeX)

y = random.randrange(\*rangeY)

if (x,y) in excluded: continue

randPoints.append((x,y))

i += 1

excluded.update((x+dx, y+dy) for (dx,dy) in deltas)

return randPoints

def sortPoints(L):

''' Mengurutkan kumpulan titik berdasarkan nilai absis yang menaik,

jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik'''

# Untungnya Python menyediakan fungsi sorted hehehe

return sorted(L, key = lambda k: [k[0], k[1]])

# Fungsi untuk Copy List

def copyList(L):

return [el for el in L]

# Fungsi untuk memeriksa apakah sebuah titik P3 berada di sebelah kiri atau kanan suatu garis yang dibentuk dua titik P1 dan P2

# return 1 jika di kiri, 0 jika pada garis, -1 jika di kanan garis

def findSide(P1, P2, P3):

val = (P1[0] \* P2[1]) + (P3[0] \* P1[1]) + (P2[0] \* P3[1]) - (P3[0] \* P2[1]) - (P2[0] \* P1[1]) - (P1[0] \* P3[1])

if val > 0:

return 1

elif val < 0:

return -1

else:

return 0

# Fungsi mengembalikan jarak titik ke garis

def lineDist(P1, P2, P3):

return abs ((P3[1] - P1[1]) \* (P2[0] - P1[0]) - (P2[1] - P1[1]) \* (P3[0] - P1[0]))

# Prosedur untuk menampilkan plot graph

def drawGraph(L1, L2, L3):

x\_val1 = [x[0] for x in L1]

y\_val1 = [x[1] for x in L1]

x\_val2 = [y[0] for y in L2]

y\_val2 = [y[1] for y in L2]

x\_val3 = [z[0] for z in L3]

y\_val3 = [z[1] for z in L3]

plt.plot(x\_val1,y\_val1,'or')

plt.plot(x\_val2,y\_val2)

plt.plot(x\_val3,y\_val3)

plt.show()

# Fungsi untuk menentukan elemen-elemen Convex Hull

def quickHull(S, p1, pn, side):

idx = -1

max\_dist = 0

i = 0

for p in S:

temp = lineDist(p1, pn, p);

if temp > max\_dist and findSide(p1, pn, p) == side:

max\_dist = temp

idx = i

i = i + 1

# Tidak ada titik yang ditemukan

if idx == -1 :

if side == 1:

left\_hull.append(p1)

return

if side == -1:

right\_hull.append(p1)

return

quickHull(S, p1, S[idx], findSide(S[idx], p1, pn))

quickHull(S, S[idx], pn, -findSide(S[idx], pn, p1))

# Program utama

if \_\_name\_\_ == '\_\_main\_\_':

n = 0

while(n <= 1):

n = input(">> Masukkan banyak titik : ")

start\_time = time.time()

# Randomisasi

Points = generatePoints(n)

# Urutkan himpunan titik

SortedPoints = copyList(sortPoints(Points))

print ">> Himpunan semesta titik : ", SortedPoints

# Titik ekstrem

p1 = SortedPoints[0]

pn = SortedPoints[len(SortedPoints)-1]

# Set hull

left\_hull = [pn]

right\_hull = [pn]

# Menentukan himpunan hull sebelah kiri

quickHull(SortedPoints, p1, pn, 1)

# Menentukan himpunan hull sebelah kanan

quickHull(SortedPoints, p1, pn, -1)

# Mengurutkan himpunan hull

upper\_hull = copyList(sortPoints(left\_hull))

lower\_hull = copyList(sortPoints(right\_hull))

# Menggabungkan hull menjadi himpunan convex hull

convex\_hull = sortPoints(list(set(upper\_hull + lower\_hull)))

print ">> Himpunan Convex Hull : ",convex\_hull

print ">> Himpunan titik pembentuk Convex Hull bagian atas : ",upper\_hull

print ">> Himpunan titik pembentuk Convex Hull bagian bawah : ",lower\_hull

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

print(">> Elapsed time = %s seconds" % (time.time() - start\_time))

drawGraph(SortedPoints, upper\_hull, lower\_hull)

**BAB II**

**DOKUMENTASI INPUT OUTPUT PROGRAM**

S

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program dapat menerima input graf |  |  |
| 1. Program dapat menghitung lintasan terpendek |  |  |
| 1. Program dapat menampilkan lintasan terpendek serta jaraknya |  |  |
| 1. Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta |  |  |