Tugas Kecil 2 IF2211 – Strategi Algoritma

Laporan

Penyelesaian Persoalan Convex Hull dengan Divide and Conquer (Quick Hull)

oleh

WILDAN DICKY ALNATARA 13516012

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2018

Algoritma dan Pseudo-code Quick Hull

Untuk mendapatkan semua titik terluar dengan quick hull kita akan menggunakan strategi divide and qonquer. Idenya adalah membagi area menjadi dua bagian dengan memilih titik paling luar dahulu. Lalu melakukan scanning ke semua titik untuk mengecek titik mana yang jaraknya paling jauh dengan garis yang dibentuk oleh 2 titik terjauh tadi dan sisinya sama dengan sisi yang sedang kita cek. Lalu lakukan dengan rekursif lagi dengan titik terjauh tersebut dengan salah satu titik terjauh dari 2 titik sebelumnya sampai hanya tersisa 2 titik dengan tidak ada titik yang berada pada sisi yang dipilih.

```
function quickHull(self,sideNow,p1,p2):
    n  = len(self.pointCollection)
    idxMax = -1
    maxJarak = 0
    #mencari titik terjauh dari garis dengan side yang sama
    for idx in range(0,n):
        panjang = self.jarakTitikGaris(p1,p2,self.pointCollection[idx])
        if self.findSide(p1,p2,self.pointCollection[idx]) ==sideNow and panjang>maxJarak:
            idxMax = idx
            maxJarak = panjang
    if idxMax ==-1: #basecase ya kalo sudah gak ada lagi titik dengan side yang sama
        self.Hull.append((p1,p2))
        return 0
    else: # lakukan qucikhull lagi
        self.__quickHull(-
        1*self.findSide(self.pointCollection[idxMax],p1,p2),self.pointCollection[idxMax],p1)
        self.__quickHull(-
        1*self.findSide(self.pointCollection[idxMax],p2,p1),self.pointCollection[idxMax],p2)
        return 0
```

Kompleksitas

Karena Algoritma ini mengadopsi cara yang dilakukan oleh quicksort maka kompleksitas big o nya yaitu O(Nlog N) dan untuk worst casenya $O(n^2)$.

Kode Program - Python

Main.py

```python
import hullAlgo
import plotWil
import random
import time


#Inisiasi
hull =  hullAlgo.convexHull()
plotting = plotWil.memPlotHull()
#Masukan input
n = int (input("Masukan jumlah titik : "))
```

```
12 lbound = int(input("Masukan batas bawah acak : "))
13 ubound =  int(input("Masukan batas atas acak : "))
14 #Proses randomized
15 print("Titik-Titiknya {x,y} : ")
16 for i in range(0,n):
17     x = random.randint(lbound,ubound)
18     y = random.randint(lbound,ubound)
19     print({x,y})
20     hull.addPoint((x,y))
21     plotting.addNotHull((x,y))
22
23 #Mulai algonya
24 msecawal = int(round(time.time() * 1000))
25 hull.mulaiHull()
26 msecakhir = int(round(time.time() * 1000))
27
28 #hasil hull nya
29 print("Hasilnya : ")
30 hasilHull = hull.Hull
31
32 #tampilkan hasil hullnya dan waktunya
33 print(hasilHull)
34 print(str (msecakhir-msecawal) + " millisecond")
35
36 #hasil plotting
37 for i in range(0,len(hasilHull)):
38     plotting.addPairHull(hasilHull[i])
39 plotting.showHull()
```

hullAlgo.py

```
class convexHull:
    def __init__(self):
        self.pointCollection = []
        self.Hull = []
    def addPoint(self,tup):
        self.pointCollection.append(tup)
    def findSide(self,p1,p2,p3):
        x1 = p1[0]
        x2 = p2[0]
        x3 = p3[0]
        y1 = p1[1]
        y2 = p2[1]
        y3 = p3[1]
```

```
        hasil = ((y2-y1)*x3) - ((x2-x1)*y3) + (x2*y1) - (x1*y2)
        if hasil>0:
            return 1
        elif hasil<0:
            return -1
        else:
            return 0
    def jarakTitikGaris(self,p1,p2,p3):
        x1 = p1[0]
        x2 = p2[0]
        x3 = p3[0]
        y1 = p1[1]
        y2 = p2[1]
        y3 = p3[1]
        hasil = ((y2-y1)*x3) - ((x2-x1)*y3) + (x2*y1) - (x1*y2)
        return abs(hasil)
    def __quickHull(self,sideNow,p1,p2):
        n  = len(self.pointCollection)
        idxMax = -1
        maxJarak = 0
        #mencari titik terjauh dari garis dengan side yang sama
        for idx in range(0,n):
            panjang = self.jarakTitikGaris(p1,p2,self.pointCollection[idx])
            if self.findSide(p1,p2,self.pointCollection[idx]) ==sideNow and panjang>maxJarak:
                idxMax = idx
                maxJarak = panjang
        if idxMax ==-1: #basecase ya kalo sudah gak ada lagi titik dengan side yang sama
            self.Hull.append((p1,p2))
            return 0
        else: # lakukan qucikhull lagi
            self.__quickHull(-
1*self.findSide(self.pointCollection[idxMax],p1,p2),self.pointCollection[idxMax],p1)
            self.__quickHull(-
1*self.findSide(self.pointCollection[idxMax],p2,p1),self.pointCollection[idxMax],p2)
            return 0

    def mulaiHull(self):
        #Cari max x i dan min x i
        n  = len(self.pointCollection)
        if n<2:
            return -1

        #mencari titik paling atas dan paling bawah
        idxMax = 0
        idxMin = 0
        for i in range(1,n):
            if self.pointCollection[i][1]> self.pointCollection[idxMax][1]:
                idxMax = i
            if self.pointCollection[i][1]< self.pointCollection[idxMin][1]:
                idxMin = i

        #mulai dnc hullnya
```

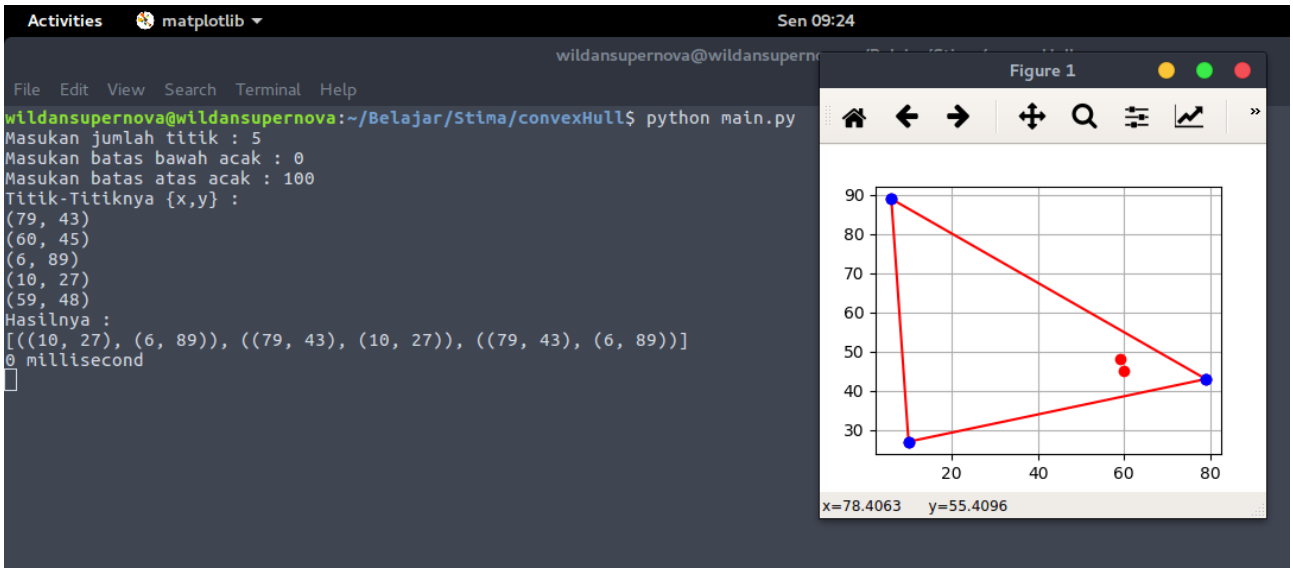
```
self.__quickHull(-1,self.pointCollection[idxMin],self.pointCollection[idxMax])
self.__quickHull(1,self.pointCollection[idxMin],self.pointCollection[idxMax])
```

plotWil.py – Visualisasi menggunakan matplotlib

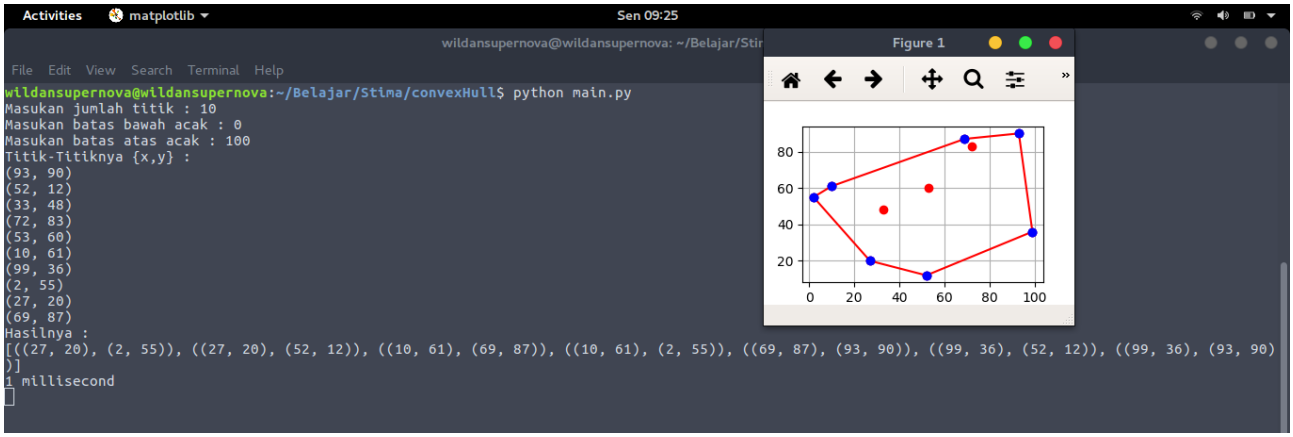
```
import matplotlib.pyplot as plt
class memPlotHull:

    def __init__(self):
        self.notHull = []
        self.pairHull = []
    def addPairHull(self,pair):
        #pair as tuple (x,y)
        self.pairHull.append(pair)
    def addNotHull(self,titik):
        #titik as tuple of tuple(x,y)
        self.notHull.append(titik)
    def showHull(self):
        nNh = len(self.notHull)
        nPh = len(self.pairHull)
        #masukan titik  not hull
        for i in range(0,nNh):
            plt.plot(self.notHull[i][0],self.notHull[i][1],'ro')
        #buat garis
        for i in range(0,nPh):
            p1 = self.pairHull[i][0]
            p2 = self.pairHull[i][1]
            plt.plot([p1[0],p2[0]],[p1[1],p2[1]],'r-')
            plt.plot([p1[0],p2[0]],[p1[1],p2[1]],'bo')
        plt.grid(True)
        plt.show()
```

Screenshot

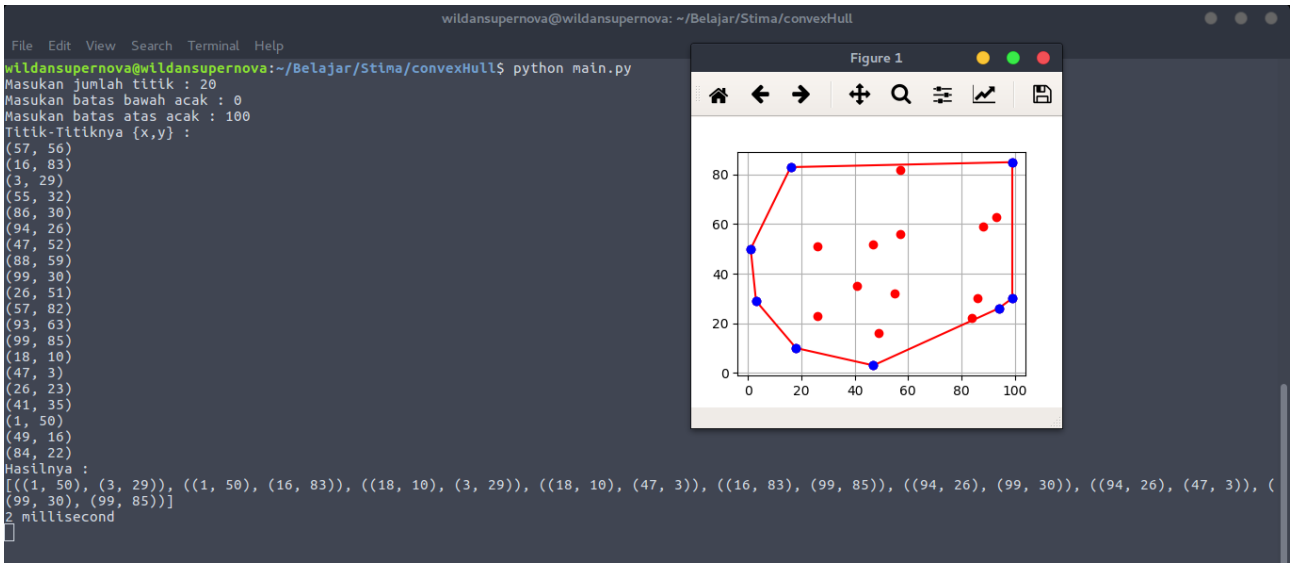
N=5



N=10



N=20



Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil running	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk semua n	√	