

IF2211 Strategi Algoritma

Algoritma Pembuatan Convex Hull dengan Divide and Queror

Tugas Kecil II



Oleh:

Muhammad Risqi Firdaus 13520043

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022**

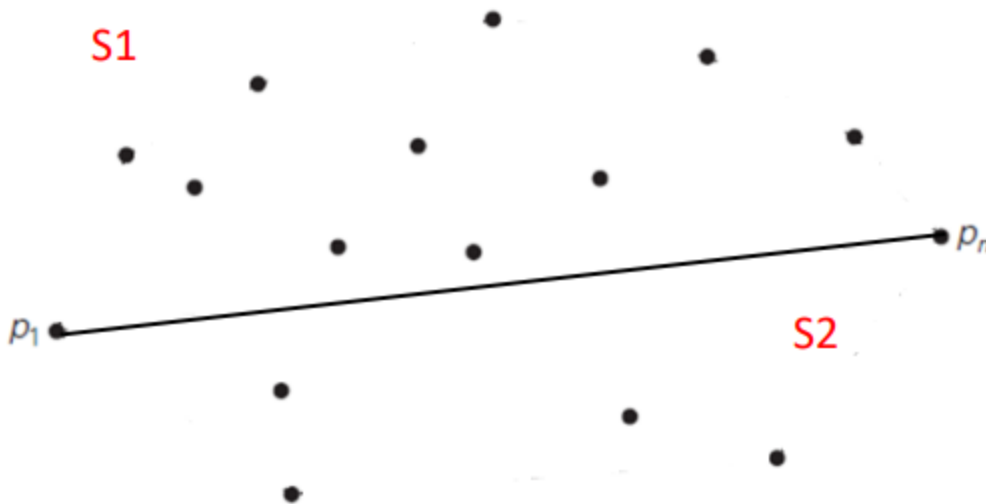
A. Penejelasan Algoritma

Convex Hull merupakan garis yang menghubungkan titik-titik terluar dari sekumpulan titik pada sebuah. Titik ya dihubungkan pasti membentuk garis conveks (cembung). Himpunan titik pada bidang planar disebut convex jika untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut.

Ada banyak algoritma yang dapat digunakan untuk menentukan convex hull, di antaranya, graham scan, jarvis, quickhull dan divide and queror. Pada kesempatan kali ini, penulis memanfaatkan algoritma divide and queror untuk menentukan himpunan titik dari convex hull.

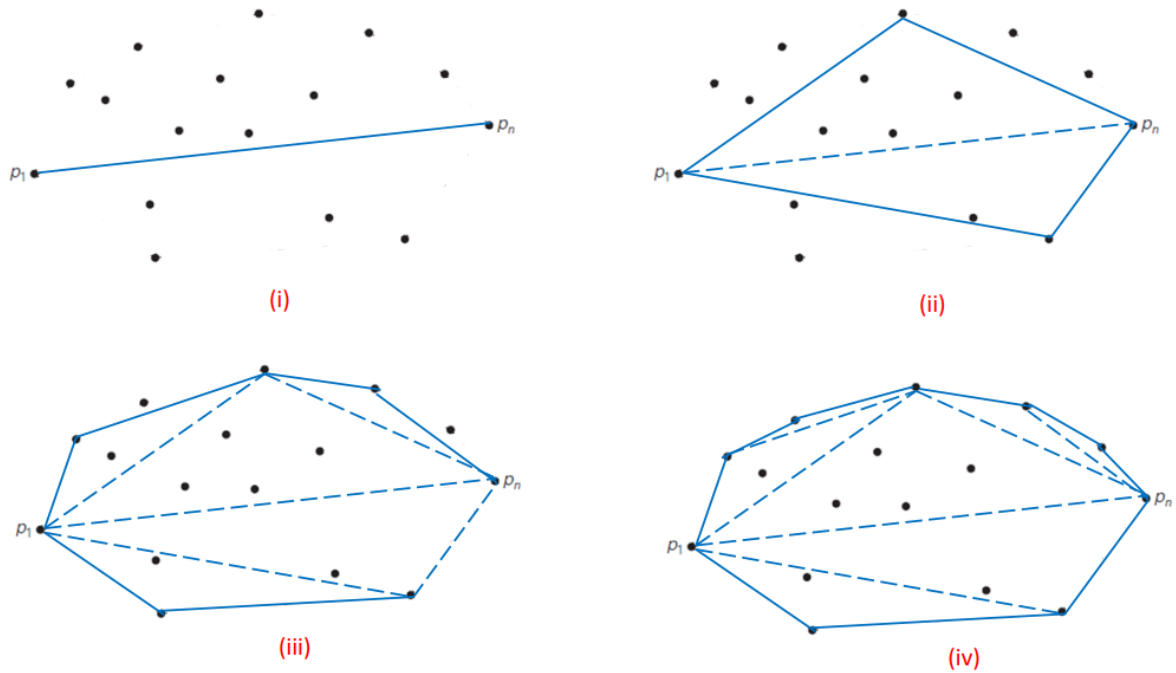
Algoritma ini memanfaatkan pengurutan quicksort untuk membantu mendapatkan titik-titik terluarnya. Komplexitasnya sebesar $O(n \log n)$. Ide dasarnya ialah, selalu menjadi membagi dua ruang bagi setiap periode kerja, ruang hasil dan ruang pembuangan.

Langkah pertama setelah mengurutkan titik-titik berdasarkan titik absisnya (sumbu-x). Ambil 2 titik dengan titik x terbesar dan terkecil, lalu buatlah garis yang membagi seluruh titik ke dalam dua bagian, bagian atas dan bagian bawah. Kemudian, untuk tiap bagian carilah titik terjauh.



Gambar 1.1 Pembagian titik oleh garis utama

Kemudian ambil titik terjauh dari tiap bagian. Pada tiap bagian bagi kembali menjadi ruang pembuangan dan ruang dari dua buah titik yang ditarik, yakni dari titik x-terkecil ke titik terjauh dari garis, dan titikx-terbesar ke titik terjauh dari garis awal. Lakukan terus menerus hingga tersisa satu titik yang masuk klasifikasi atau tidak tersisa titik sama sekali.



Gambar 1.2 Visualisasi pengambilan titik pada tiap periode

Terkahir gabungkan hasil dari tiap fungsi, fungsi atas kanan-kiri, serta fungsi bawah kanan kiri. Sehingga terbentuklah himpunan titik terluar convex hull. Untuk penyelesaian, dapat dilakukan plot scatter untuk semua himpunan titik awal dan plot garis untuk titik terluar, untuk melihat pembentukan convex hull dari dataset.

B. Kode Program

a. Himpunan Fungsi Pada Modul

No	Nama Fungsi	Penjelasan
1.	Def __init__(self, pairOfSeries)	Constructor dari object ConvexHull
2.	Def fetPoints(self, mostLeft, mostRight, listOfP, isTop)	Fungsi untuk mencari titik terjauh dengan rekursi dan divide and queror.
3.	Def myConvexHull(self, P)	Fungsi yang mengurutkan himpunan titik berdasarkan sumbu-x menaik, serta memecah berdasarkan garis utama dan mengembalikannya dalam bentuk numpy array
	Def getDetOf3Points(self, p1, p2, p)	Fungsi mengembalikan nilai determinan (jarak) dari sebuah titik p dan sebuah garis

	yang dibentuk dua titik p1 dan p2.
--	------------------------------------

b. Tangkapan Layar untuk Program

main.py

```
from MyConvexHull import myConvexHull
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

print("Pilih dataset: ")
print("1. Iris")
print("2. Breast Cancer")
print("3. Wine")
a = int(input("Masukkan nomor data: "))
if(a==1):
    data = datasets.load_iris()
elif(a==2):
    data = datasets.load_breast_cancer()
elif (a==3):
    data=datasets.load_wine()

#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)

#figuring
plt.figure(figsize = (10, 6))
colors = ['b','r','g']

#choose feature
for i in range(0,len(data.feature_names)//2+1,2):
    print(i//2+1,"{} vs {}".format(data.feature_names[i],data.feature_names[i+1]))
n=int(input("Masukkan nomor data yang mau dianalisis: "))

#make plot
n=2*(n-1)
plt.title('{} vs {}'.format(data.feature_names[n],data.feature_names[n+1]));
plt.xlabel(data.feature_names[n])
plt.ylabel(data.feature_names[n+1])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[n,n+1]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range (len(hull.ver)):
        plt.plot([hull.ver[j][0],hull.ver[(j+1)%len(hull.ver)][0]] , [hull.ver[j][1],hull.ver[(j+1)%len(hull.ver)][1]] ,c =colors[i])
plt.legend()
plt.show()
```

myConvexHull.py

```

import numpy as np
class myConvexHull:
    def __init__(self, pairOfSeries):
        self.ver = self.ConvexHull(pairOfSeries)
        self.full = pairOfSeries

    def ConvexHull(self, P):
        P = sorted(P, key=lambda x: x[0])
        a = [P[0]] + self.getPoints(P[0], P[-1], P[1:-1], False) + [P[-1]] + self.getPoints(P[0], P[-1], P[1:-1], True)
        s = np.array(a)
        return s

```

```

def getPoints(self, mostLeft, mostRight, listOfP, isTop):
    if (len(listOfP) <= 1):
        if (len(listOfP) == 0):
            return []
        elif ((self.getDetOf3Points(mostLeft, mostRight, listOfP[0]) > 0 and not isTop) or (self.getDetOf3Points(mostLeft, mostRight, listOfP[0]) < 0 and isTop)):
            return []
        else:
            return [listOfP[0]]
    idx = -1
    maxDist = 0
    for i in range(len(listOfP)):
        temp = self.getDetOf3Points(mostLeft, mostRight, listOfP[i])
        if (temp < maxDist and not isTop):
            idx = i
            maxDist = temp
        elif (temp > maxDist and isTop):
            idx = i
            maxDist = temp
    if (idx == -1):
        return []
    if (isTop):
        right = self.getPoints(listOfP[idx], mostRight, listOfP[idx+1:], True)
        left = self.getPoints(mostLeft, listOfP[idx], listOfP[0:idx], True)
        return right + [listOfP[idx]] + left
    else:
        left = self.getPoints(mostLeft, listOfP[idx], listOfP[0:idx], False)
        right = self.getPoints(listOfP[idx], mostRight, listOfP[idx+1:], False)
        return left + [listOfP[idx]] + right

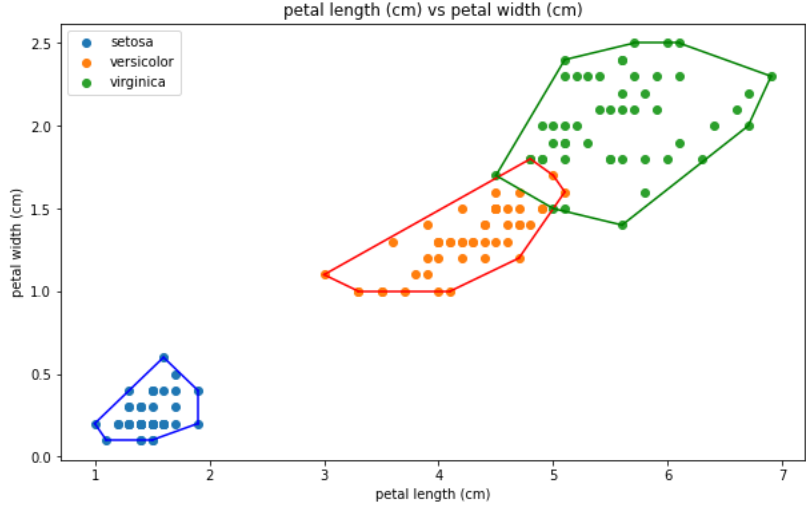
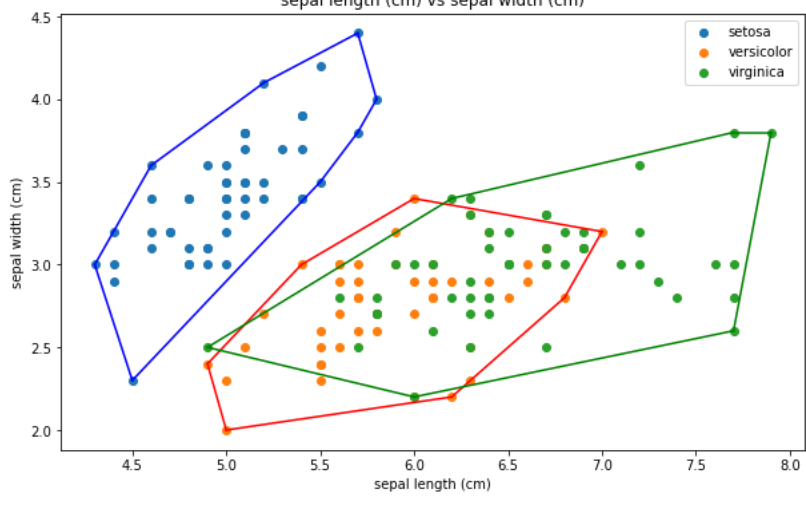
```

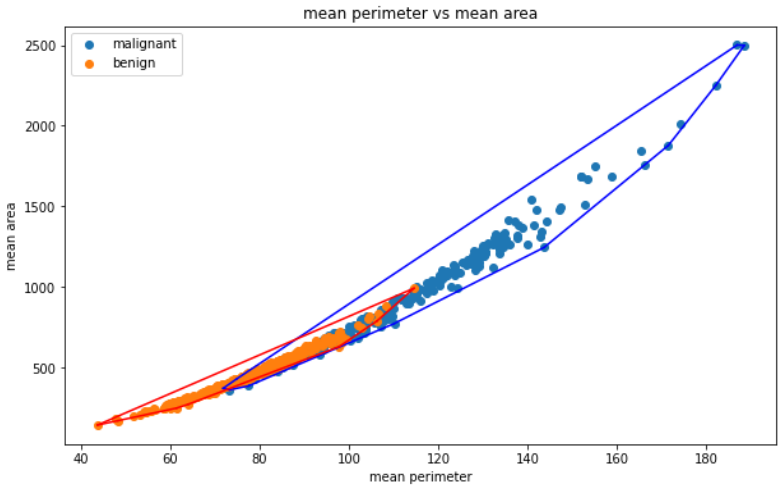
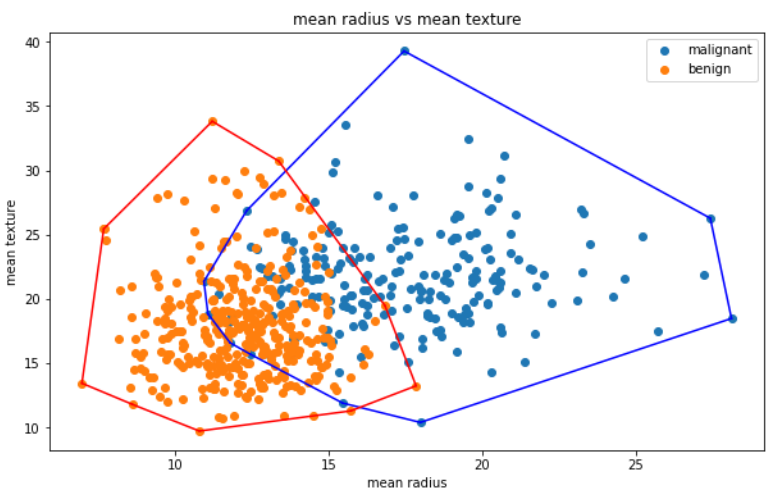
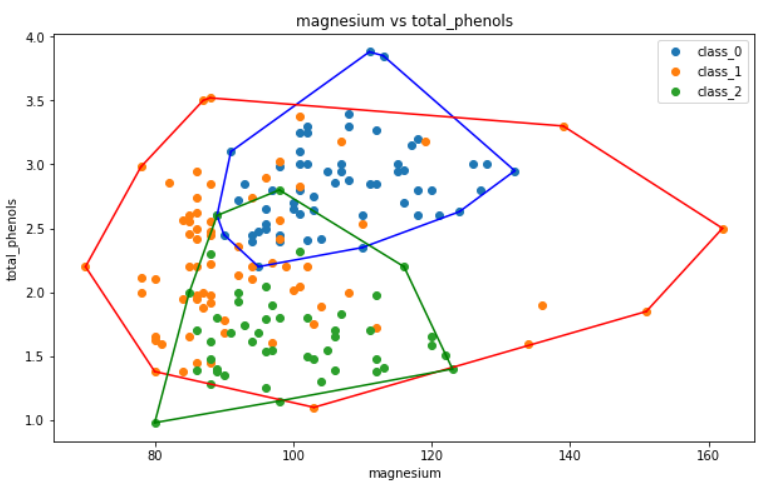
```

def getDetOf3Points(self, p1, p2, p):
    nilai_kanan = p1[0]*p2[1] + p[0]*p1[1] + p2[0]*p[1]
    nilai_kiri = p[0]*p2[1] + p2[0]*p1[1] + p1[0]*p[1]
    return nilai_kanan - nilai_kiri

```

C. Hasil dari Program

No	Nama Dataset	Hasil
1.	Iris (Petal Width vs Petal Length)	 <p>petal length (cm) vs petal width (cm)</p> <p>Legend: setosa (blue), versicolor (orange), virginica (green)</p>
2.	Iris (Sepal Width vs Sepal Length)	 <p>sepal length (cm) vs sepal width (cm)</p> <p>Legend: setosa (blue), versicolor (orange), virginica (green)</p>

3.	Breast cancer (mean perimeter vs mean area)	 <p>Scatter plot titled "mean perimeter vs mean area" showing mean area (y-axis, 0 to 2500) versus mean perimeter (x-axis, 40 to 180). The plot compares malignant (blue dots) and benign (orange dots) breast cancer samples. Both classes show a strong positive linear correlation between mean perimeter and mean area. Malignant samples generally have higher values for both metrics compared to benign samples.</p>
4.	Breast cancer (mean radius vs mean texture)	 <p>Scatter plot titled "mean radius vs mean texture" showing mean texture (y-axis, 10 to 40) versus mean radius (x-axis, 10 to 25). The plot compares malignant (blue dots) and benign (orange dots) breast cancer samples. Benign samples are clustered at lower values for both mean radius and mean texture. Malignant samples are more spread out, generally having higher mean texture values across a range of mean radii.</p>
5.	Wine (Magnesium vs total phenols)	 <p>Scatter plot titled "magnesium vs total_phenols" showing total phenols (y-axis, 1.0 to 4.0) versus magnesium (x-axis, 80 to 160). The plot compares three wine classes: class_0 (blue dots), class_1 (orange dots), and class_2 (green dots). Class_1 is clustered at higher magnesium and total phenols values. Class_2 is clustered at lower values for both metrics. Class_0 is more spread out in the middle range of both variables.</p>

Link: https://github.com/mrfirdauss-20/Tucil-Stima2022/tree/main/Tucil2_13520043