**TUGAS KECIL**

**Implementasi Algoritma Divide and Conquer dalam Pencarian Convex Hull sebagai Pengujian Linear Separability**

**LAPORAN**

**Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada**

**Semester II**

**Tahun Akademik 2021-2022**

**oleh**

**Owen Christian Wijaya 13520124**

**A picture containing text

Description automatically generated**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2022**

DAFTAR ISI

[BAB I. ALGORITMA *DIVIDE AND CONQUER* 2](#_Toc96675269)

[BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON 4](#_Toc96675270)

[2.1 process.py 4](#_Toc96675271)

[2.2 utils.py 7](#_Toc96675272)

[BAB III. PENGUJIAN (INPUT/OUTPUT) 9](#_Toc96675273)

[IV. *REPOSITORY* 10](#_Toc96675274)

# BAB I. ALGORITMA *DIVIDE AND CONQUER*

Dalam tugas kecil ini, algoritma *divide and conquer* digunakan untuk menemukan titik-titik dalam suatu kumpulan data dua-dimensi (x, y) yang membentuk sebuah *convex hull*. *Convex hull* adalah sebuah bangun datar cembung yang dibangun mengelilingi kumpulan data titik dalam suatu *scatter plot. Convex hull* umumnya digunakan untuk meninjau apakah dua buah set data dapat dipisahkan dengan suatu garis linear (*linear separability*). Dua set data disebut *linearly separable* (dapat dipisahkan secara linear) apabila *convex hull* yang dibentuk oleh kedua data tersebut tidak beririsan.

Algoritma yang digunakan diterapkan dalam *class* Convex, fungsi ConvexHull. Fungsi ConvexHull menerima suatu list berisi koordinat (x, y) dan mengubah list tersebut ke sebuah list berisi ADT Point. Fungsi akan mengurutkan list tersebut berdasarkan koordinat absis sehingga dapat ditemukan titik dengan absis minimum (minAbs) dan maksimum (maxAbs). Selanjutnya, fungsi akan membagi titik-titik yang tersisa ke bagian kiri dan kanan garis tergantung determinan titik tersebut dengan titik absis minimum dan maksimum menggunakan persamaan berikut:

Titik p akan dikategorikan sebagai titik di sebelah kiri garis apabila hasil determinan positif. Sebaliknya, titik p akan dikategorikan berada di kanan garis apabila hasil determinan negatif. Apabila hasil determinan 0, maka titik tersebut berada di garis yang sama dengan garis yang dibentuk titik absis minimum dan maksimum, sehingga titik tersebut tidak dapat membentuk suatu *convex hull* dan tidak perlu dikategorikan.

Selanjutnya, fungsi akan mencari titik terjauh (pMax) dari bagian kiri dan bagian kanan. Proses pembagian di sisi kiri dilakukan pada fungsi divideLeft, sementara proses pembagian di si kanan dilakukan pada fungsi divideRight. Kedua fungsi ini bersifat rekursif, dengan basis apabila list kosong maka proses rekursi akan dihentikan. Sebaliknya, fungsi akan mencari titik terjauh dari garis tersebut menggunakan fungsi findPMax. Fungsi akan mencari jarak antara garis dengan titik menggunakan persamaan:

Ditemukan persamaan garis dengan Jarak antar garis dapat dihitung dengan persamaan . Apabila ada dua titik dengan jarak yang sama, maka pMax ditentukan berdasarkan titik yang membuat sudut terbesar antara pMax, minAbs dan maxAbs menggunakan aturan cosinus . pMax akan ditambahkan ke dalam list titik, dan kemudian akan dilakukan proses rekursif (divideLeft/divideRight tergantung sisi) dengan membagi titik-titik yang tersisa ke sisi kiri / kanan dari garis yang dibentuk titik minAbs dan pMax serta garis pMax dan maxAbs. Proses ini akan terus berulang sampai tidak ada titik yang dapat dievaluasi lagi.

Setelah proses rekursi selesai, fungsi akan menggabungkan hasil titik-titik dengan mengurutkan titik pada bagian kiri secara terurut membesar berdasarkan absis, dan mengurutkan titik pada bagian kanan secara terurut mengecil berdasarkan absis, Fungsi akan mengembalikan dua buah list berisi daftar koordinat absis dan ordinat yang berkorespondensi. Kedua array dimasukkan sebagai parameter pada plt.plot untuk menggambarkan *convex hull*.

# BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON

## 2.1 process.py

(file utama berisi fungsi *divide and conquer*)

import myConvexHull.utils as ut

class Convex(object):

    def \_\_init\_\_(self):

        self.PointsList = []

    def divideList(self, PointsList, minAbs, maxAbs, flag):

        '''

        FUNGSI DIVIDE:

        Membagi list of points menjadi dua bagian berdasarkan determinan

        relatif terhadap garis yang dibentuk titik absis minimum dan maksimum.

        Titik yang mempunyai determinan positif terhadap garis dimasukkan ke

        list leftSide, sementara titik dengan determinan negatif dimasukkan ke

        list rightSide.

        Argumen fungsi:

            - PointsList: list of points yang akan dibagi

            - minAbs, maxAbs: titik dengan absis minimum/maksimum

            - flag: integer untuk menentukan list mana yang akan dikembalikan

            (beberapa kasus membutuhkan hanya list kiri / kanan)

        '''

        leftSide = []

        rightSide = []

        for i in range(len(PointsList)):

            if (((PointsList[i].x > minAbs.x) or (PointsList[i].x == minAbs.x and PointsList[i].y < minAbs.y))

                and ((PointsList[i].x < maxAbs.x) or (PointsList[i].x == maxAbs.x and PointsList[i].y < maxAbs.y))):

                if (ut.isDeterminantPositive(minAbs, maxAbs, PointsList[i])):

                    leftSide.append(PointsList[i])

                if (not ut.isDeterminantPositive(minAbs, maxAbs, PointsList[i])):

                    rightSide.append(PointsList[i])

        if (flag > 0):

            return leftSide

        elif (flag < 0):

            return rightSide

        else:

            return leftSide, rightSide

    def divideLeft(self, PointsList, minAbs, maxAbs):

        '''

        Membagi list of points menjadi dua bagian berdasarkan determinan

        untuk bagian kiri dari garis awal. Fungsi ini akan melakukan

        pembagian secara rekursif sampai list kosong.

        Argumen fungsi:

            - PointsList: list of points yang akan dibagi

            - minAbs, maxAbs: titik dengan absis minimum/maksimum

        '''

        temp = []

        if len(PointsList) == 0:

            return []

        else:

            pMax = ut.findPMax(PointsList, minAbs, maxAbs)

            PointsList.remove(pMax)

            temp.append(pMax)

            leftTemp = self.divideLeft(self.divideList(PointsList, minAbs, pMax, 1), minAbs, pMax)

            if (len(leftTemp) > 0):

                for point in leftTemp:

                    PointsList.remove(point)

                    temp.append(point)

            rightTemp = self.divideLeft(self.divideList(PointsList, pMax, maxAbs, 1), pMax, maxAbs)

            if (len(rightTemp) > 0):

                for point in rightTemp:

                    PointsList.remove(point)

                    temp.append(point)

            return temp

    def divideRight(self, PointsList, minAbs, maxAbs):

        '''

        Membagi list of points menjadi dua bagian berdasarkan determinan

        untuk bagian kanan dari garis awal. Fungsi ini akan melakukan

        pembagian secara rekursif sampai list kosong.

        Argumen fungsi:

            - PointsList: list of points yang akan dibagi

            - minAbs, maxAbs: titik dengan absis minimum/maksimum

        '''

        temp = []

        if len(PointsList) == 0: #basis apabila list kosong

            return []

        else:

            pMax = ut.findPMax(PointsList, minAbs, maxAbs)

            PointsList.remove(pMax)

            temp.append(pMax)

            leftTemp = self.divideRight(self.divideList(PointsList, minAbs, pMax, -1), minAbs, pMax)

            if (len(leftTemp) > 0):

                for point in leftTemp:

                    PointsList.remove(point)

                    temp.append(point)

            rightTemp = self.divideRight(self.divideList(PointsList, pMax, maxAbs, -1), pMax, maxAbs)

            if (len(rightTemp) > 0):

                for point in rightTemp:

                    PointsList.remove(point)

                    temp.append(point)

            return temp

    def mergeList(self, leftRes, rightRes, minAbs, maxAbs):

        '''

        Menggabungkan titik yang berada di dalam list leftRes dan rightRes

        (hasil divide and conquer) dengan titik absis minimum/maksimum.

        Argumen fungsi:

            - leftRes, rightRes: hasil dari divideLeft dan divideRight

            - minAbs, maxAbs: titik dengan absis minimum/maksimum

        '''

        mergedList = []

        # sort leftRes ascending, sort rightRes descending

        for i in range(len(leftRes) - 1):

            for j in range(len(leftRes) - i - 1):

                if (leftRes[j].x > leftRes[j + 1].x):

                    leftRes[j], leftRes[j + 1] = leftRes[j + 1], leftRes[j]

        for i in range(len(rightRes) - 1):

            for j in range(len(rightRes) - i - 1):

                if (rightRes[j].x < rightRes[j + 1].x):

                    rightRes[j], rightRes[j + 1] = rightRes[j + 1], rightRes[j]

        mergedList.append(minAbs)

        for i in range(len(leftRes)):

            mergedList.append(leftRes[i])

        mergedList.append(maxAbs)

        for i in range(len(rightRes)):

            mergedList.append(rightRes[i])

        mergedList.append(mergedList[0])

        mergedX = []

        mergedY = []

        for i in range(len(mergedList)):

            mergedX.append(float(mergedList[i].x))

            mergedY.append(float(mergedList[i].y))

        return [mergedX, mergedY]

    def ConvexHull(self, listOfPoints):

        '''

        Fungsi utama (saat pemanggilan)

        Menerima dataset dalam bentuk list dua dimensi (x, y)

        '''

        PointsList = []

        # membuat list yang berisi objectPoint

        for i in range(len(listOfPoints)):

            PointsList.append(ut.Point(listOfPoints[i][0], listOfPoints[i][1]))

        # bubble sort berdasarkan koordinat x

        for i in range(len(PointsList) - 1):

            for j in range(len(PointsList) - i - 1):

                if (PointsList[j].x > PointsList[j + 1].x):

                    PointsList[j], PointsList[j + 1] = PointsList[j + 1], PointsList[j]

        # mencari nilai absis minimum dan maksimum untuk

        # mendapatkan garis yang membagi points menjadi 2 bagian

        minAbs = PointsList[0]

        maxAbs = PointsList[len(PointsList) - 1]

        PointsList.remove(minAbs)

        PointsList.remove(maxAbs)

        leftSide, rightSide = self.divideList(PointsList, minAbs, maxAbs, 0)

        leftRes = self.divideLeft(leftSide, minAbs, maxAbs)

        rightRes = self.divideRight(rightSide, minAbs, maxAbs)

        return self.mergeList(leftRes, rightRes, minAbs, maxAbs)

2.2 utils.py   
(file berisi fungsi utilitas untuk membantu fungsi utama)

import math

class Point:

    '''

    Class Point untuk merepresentasikan sebuah titik

    dalam bidang dua-dimensi.

    '''

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

def isDeterminantPositive(p1, p2, p3):

    '''

    Menghitung determinan antara dua buah titik (p1 dan p2)

    yang membentuk sebuah garis dan titik p3 untuk menentukan

    letak sebuah titik relatif terhadap garis.

    - Determinan positif: titik berada pada sisi kiri relatif dari garis

    - Determinan negatif: titik berada pada sisi kanan relatif dari garis

    - Determinan nol: titik berada pada garis dan dapat diabaikan

    Argumen fungsi:

        - p1, p2: titik yang membentuk garis (absis minimum dan maksimum)

        - p3: titik yang dibandingkan

    '''

    det = ((p1.x \* p2.y) + (p3.x \* p1.y) + (p2.x \* p3.y)

            - (p3.x \* p2.y) - (p1.x \* p3.y) - (p2.x \* p1.y))

    if (det > 0):

        return True

    if (det < 0):

        return False

def distance(p1, p2, p3):

    '''

    Menghitung jarak antara garis yang dibentuk p1 dan p2

    dengan titik p3.

    Argumen fungsi:

        p1, p2: titik yang membentuk garis (absis minimum dan maksimum)

        p3: titik yang dibandingkan

    '''

    A = p1.y - p2.y

    B = p2.x - p1.x

    C = p1.x \* p2.y - p2.x \* p1.y

    dist = abs((A \* p3.x + B \* p3.y + C) / ((A \*\* 2 + B \*\* 2) \*\* 0.5))

    return dist

def findAngle(minAbs, maxAbs, pMax):

    '''

    Mencari sudut dari dari sudut yang dibentuk titik pMax, minAbs

    dan maxAbs menggunakan aturan cosinus

    Argumen fungsi:

        minAbs, maxAbs, pMax: titik-titik sebagai parameter

    '''

    aSide = (pMax.x - maxAbs.x) \*\* 2 + (pMax.y - maxAbs.y) \*\* 2

    bSide = (pMax.x - minAbs.x) \*\* 2 + (pMax.y - minAbs.y) \*\* 2

    cSide = (maxAbs.x - minAbs.x) \*\* 2 + (maxAbs.y - minAbs.y) \*\* 2

    cos = (aSide - bSide - cSide) / (-2 \* (bSide \*\* 0.5) \* (cSide \*\* 0.5))

    return math.acos(cos) \* 180 / math.pi

def findPMax(PointsList, minAbs, maxAbs):

    '''

    Fungsi untuk menentukan pMax, titik berjarak terjauh dari garis

    yang dibentuk minAbs dan maxAbs.

    Argumen fungsi:

        - PointsList: list of points yang akan dibagi

        - minAbs, maxAbs: titik dengan absis minimum/maksimum

    '''

    currentDistance = 0

    maxDistance = 0

    index = 0

    for i in range(len(PointsList)):

        currentDistance = distance(minAbs, maxAbs, PointsList[i])

        if currentDistance > maxDistance or (currentDistance == maxDistance and

            findAngle(minAbs, maxAbs, PointsList[i]) > findAngle(minAbs, maxAbs, PointsList[index])):

            maxDistance = currentDistance

            index = i

    return PointsList[index]

## 2.3 main.py

(penggunaan pencarian Sepal Length vs Sepal Width pada dataset iris)

#%%

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

from myConvexHull.process import Convex

from scipy.spatial import ConvexHull

#%%

title = "Sepal Length vs Sepal Width"

data = datasets.load\_iris()

df = pd.DataFrame(data.data, columns = data.feature\_names)

df['Target'] = pd.DataFrame(data.target)

df.head()

#%%

# Penggunaan library myConvexHull

plt.figure(figsize = (10, 6))

plt.title(title + ' (myConvexHull)')

plt.xlabel(data.feature\_names[0])

plt.ylabel(data.feature\_names[1])

ConvexObj = Convex()

colors = ['b', 'r', 'g']

for i in range(len(data.target\_names)):

    bucket = df[df['Target'] == i].iloc[:, [0, 1]].values

    hull = ConvexObj.ConvexHull(bucket)

    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i], color = colors[i % len(colors)])

    plt.plot(hull[0], hull[1], color = colors[i % len(colors)])

plt.legend()

#%%

# Perbandingan menggunakan library SciPy

plt.figure(figsize = (10, 6))

plt.title(title + ' (SciPy)')

plt.xlabel(data.feature\_names[0])

plt.ylabel(data.feature\_names[1])

scolors = ['c', 'm', 'y']

for i in range(len(data.target\_names)):

    bucket = df[df['Target'] == i].iloc[:, [0, 1]].values

    hull = ConvexHull(bucket)

    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i], color = scolors[i % len(scolors)])

    for simplex in hull.simplices:

        plt.plot(bucket[simplex, 0], bucket[simplex, 1], color = scolors[i % len(scolors)])

plt.legend()

# BAB III. PENGUJIAN (INPUT/OUTPUT)

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan (no syntax error) | √ |  |
| 2. Program berhasil running | √ |  |
| 3. Program dapat membaca file masukan dan menuliskan luaran | √ |  |
| 4. Program berhasil menemukan semua kata di dalam puzzle | √ |  |

# BAB IV. *REPOSITORY*

*Repository* dapat diakses via https://github.com/clumsyyyy/TucilStima1 (branch main)