Untuk Memenuhi Tugas Mata Kuliah

Strategi Algoritma (IF 2211)

Nama : Jaya Mangalo Soegeng Rahardjo

NIM : 13520015

TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. ALGORITMA PROGRAM

Step 1: Divide Half

Program akan mencari dot paling kiri secara x dan dot paling kanan secara x. Kedua titik ini akan digunakan untuk membuat garis yang memisahkan bagian atas dan bagian bawah.

Program akan memisahkan array semua dot menjadi 2 array yaitu array atas dan array bawah. Kedua bagian tersebut akan dipisah lagi di step selanjutnya

Step 2: Recursive Divide

Program akan diberi parameter dot-kiri dan dot-kanan dan membagi array dot bagian atas dan array dot bagian bawah secara rekursif.

Basis rekursif adalah jika array kosong, program akan me-return pasangan dot yang akan menjadi bagian dari convex hull.

Rekursifnya adalah program akan mencari dot yang paling jauh dari garis yang dibaut titik dot-kiri dan dot-kanan, Setelah itu, program akan mencari titik-titik yang diluar garis yang dibuat oleh dot-kiri, dot-jauh, serta titik-titik yang diluar garis dot-jau, dot kanan. Dot-dot diluar segitiga tersebut akan dimasukkan lagi kedalam pembagian rekursif sampai habis.

Step 3: Combine

Setelah selesai rekusifnya, program akan menggabungkan semua dot-dot yang merupakan hasil dari program rekursif, hasilnya adalah array of array dari titik-titik yang membuat convex hull.

B. SOURCE CODE

Setup and global variables

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "pcolor.h"

char P[100][100];       //matrix of chars
int colour[100][100];  //matrix of chars that should be colored

char K[100][100];

int x;
int y;

int comparisoncount=0;
int currentword=1;
int word=0 ; //banyak word (+1 dari \n)

```

Function Main

src > main.py > ...

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

data = datasets.load_iris()

#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)

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

print(df)

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = CONVEXHULL(bucket) #bagian ini diganti dengan hasil implementasi

    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[1])

    for j in range(0,len(hull),2):
        x_values = [hull[j][0], hull[j+1][0]]
        y_values = [hull[j][1], hull[j+1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()

plt.show()

```

Function CONVEXHULL

```python
#Main Function:
#Find dot at extreme left and extreme right
#Using Func SplitTwo splits arr_of_dots into upperpart and lowerpart
#Uses 2 instances of Func ConvexHull2 to find the convexhull of the upper part and lowerpart
#Returns the combined result of top convexhull2 and bottom convexhull2, creating Convexhull
#Final Return structure Example [[dot1],[dot2],[dot2],[dot2],[dot3],[dot4],[dot4],[dot5],[dot5],[dot1]]
#Tip: Connect pairs of 2 dots to make a line
#Example: dot1 to dot2, dot2 to dot3, dot3 to dot4, dot4 to dot5, dot5 to dot1 to make the full convexhull
def CONVEXHULL(arr_of_dots):
    min_index_col = np.argmin(arr_of_dots, axis=0)[0]
    max_index_col = np.argmax(arr_of_dots, axis=0)[0]

    mindot = arr_of_dots[min_index_col]
    maxdot = arr_of_dots[max_index_col]

    upperpart,lowerpart = SplitTwo(mindot,maxdot,arr_of_dots)

    Convex = []
    for x in (ConvexHull2(mindot,maxdot,upperpart,True)):
        Convex.append(x)
    for x in (ConvexHull2(mindot,maxdot,lowerpart,False)):
        Convex.append(x)
    return Convex
```

Function ConvexHull2

```
#Using Func findfurthest finds point furthest from line of mindot and maxdot
#And then using Func Split split arr_of_dots into 2 areas based on bool (True means takes upper part, False means take bottom part)
#And then recursively calls itself with the parameters of 2/3 dots and the split parts.
#Returns the mindot and maxdot when arr_of_dots is empty
#After all recursions are done: Returns all of the dots that make up the convex
#Final Return structure Example [[dot1],[dot2],[dot2],[dot2],[dot3],[dot4]]
#Tip: Connect pairs of 2 dots to make a line
#Example: dot1 to dot2, dot2 to dot3, dot3 to dot4 to make a half finished perimeter
#The other convexhull2 will make the second half of the perimeter and combined will make the convex hull
def ConvexHull2(mindot,maxdot,arr_of_dots,bool):
    if(arr_of_dots.size != 0):
        maxdistancedot = FindFurthest(mindot,maxdot,arr_of_dots)

        part = Split(mindot,maxdistancedot,arr_of_dots,bool)
        part2  = Split(maxdistancedot,maxdot,arr_of_dots,bool)

        Convex = []
        for x in (ConvexHull2(mindot,maxdistancedot,part,bool)):
            Convex.append(x)
        for x in (ConvexHull2(maxdistancedot,maxdot,part2,bool)):
            Convex.append(x)
        return Convex
    else:
        return maxdot.tolist(),mindot.tolist()
```

Function FindFurthest

```python
#find the dot furthest from the line made by mindot and maxdot from arr_of_dots
def FindFurthest(mindot,maxdot,arr_of_dots):
    x1 = mindot[0]
    y1 = mindot[1]
    x2 = maxdot[0]
    y2 = maxdot[1]

    b = -(x2-x1)            #find a, b, c created by mindot and maxdot
    a = y2-y1
    c = -a * x1 - b*y1

    max_dist = 0
    maxangle = 0
    for dots in arr_of_dots:
        x3 = dots[0]
        y3 = dots[1]

        dist = abs(a * x3 + b * y3 + c) #check distance of dot to line, Note: division by sqrt(a^2+b^2) is not needed because it is a const

        if dist > max_dist :        #if dot's distance is further than  maxdot, replace maxdot and  maxangle
            max_dist = dist
            maxdistancedot = dots
            d12 = np.sqrt((x1-x2)**2 + (y1-y2)**2)
            d13 = np.sqrt((x1-x3)**2 + (y1-y3)**2)
            d23 = np.sqrt((x3-x2)**2 + (y3-y2)**2)
            maxangle = np.arccos((d13*d13 + d23*d23 - d12*d12)/(2*d13*d23))
        elif(dist == max_dist):     #if dot is same distance than maxdot, compare angle and maxangle. Replace maxdot and maxangle if new an
            d12 = np.sqrt((x1-x2)**2 + (y1-y2)**2)
            d13 = np.sqrt((x1-x3)**2 + (y1-y3)**2)
            d23 = np.sqrt((x3-x2)**2 + (y3-y2)**2)
            angle = np.arccos((d13*d13 + d23*d23 - d12*d12)/(2*d13*d23))
            if(angle > maxangle):
                max_dist = dist
                maxdistancedot = dots
                maxangle = angle
    return maxdistancedot
```

Function SplitTwo

```python
#Split arr_of_dots into two parts (top and bottom) and returns array of dots of top and bottom parts
def SplitTwo(mindot,maxdot,arr_of_dots):
    x1 = mindot[0]
    y1 = mindot[1]
    x2 = maxdot[0]
    y2 = maxdot[1]

    upperpart = np.array([])
    lowerpart = np.array([])

    for dots in arr_of_dots:
        x3 = dots[0]
        y3 = dots[1]
        det = round(((x1*y2)+(x3*y1)+(x2*y3)-(x3*y2)-(x2*y1)-(x1*y3)),10)   #det is rounded to 10 because float multiplication
                                                                            #sometimes has a residue 0.000000000001, causing problems
        if(det > 0):                                        #toppart
            if upperpart.size == 0:
                upperpart = np.array([dots])
            upperpart = np.vstack([upperpart,dots])
        elif(det < 0):                                      #bottompart
            if lowerpart.size == 0:
                lowerpart = np.array([dots])
            lowerpart = np.vstack([lowerpart,dots])
    return upperpart, lowerpart
```

Function Split

```
def Split(mindot,maxdot,arr_of_dots,bool):
    x1 = mindot[0]
    y1 = mindot[1]
    x2 = maxdot[0]
    y2 = maxdot[1]

    part = np.array([])

    if bool:                            #Bool = True
        for dots in arr_of_dots:
            x3 = dots[0]
            y3 = dots[1]
            det = round(((x1*y2)+(x3*y1)+(x2*y3)-(x3*y2)-(x2*y1)-(x1*y3)),10)
            if(det > 0):                        #toppart
                if part.size == 0:
                    part = np.array([dots])
                else:
                    part = np.vstack([part,dots])
    else:                                #Bool = False
        for dots in arr_of_dots:
            x3 = dots[0]
            y3 = dots[1]
            det = round(((x1*y2)+(x3*y1)+(x2*y3)-(x3*y2)-(x2*y1)-(x1*y3)),10)
            if(det < 0):                        #bottompart
                if part.size == 0:
                    part = np.array([dots])
                else:
                    part = np.vstack([part,dots])
    return part
```

C. TEST CASES

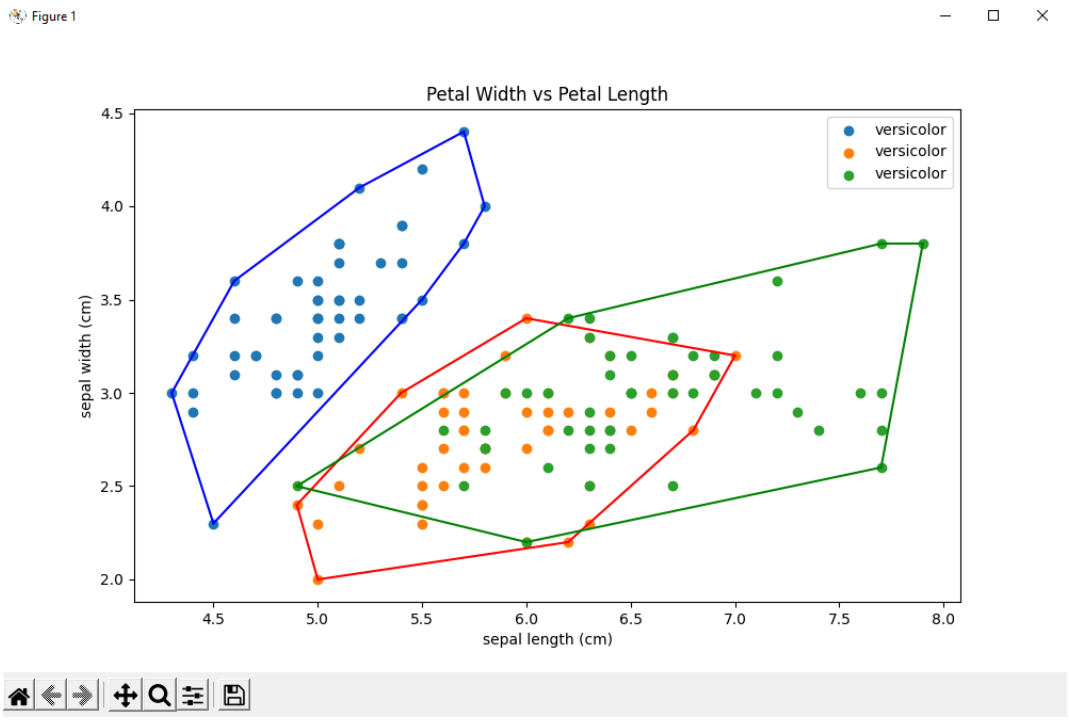
DataSet Iris

Dataset:

```
[Running] python -u "e:\Tugas Itb\Sem 4\Stima\Tucil\Tucil-2-Stima\src\main.py"
     sepal length (cm)  sepal width (cm)  ...  petal width (cm)  Target
0                  5.1               3.5  ...               0.2       0
1                  4.9               3.0  ...               0.2       0
2                  4.7               3.2  ...               0.2       0
3                  4.6               3.1  ...               0.2       0
4                  5.0               3.6  ...               0.2       0
..                 ...               ...  ...               ...     ...
145                6.7               3.0  ...               2.3       2
146                6.3               2.5  ...               1.9       2
147                6.5               3.0  ...               2.0       2
148                6.2               3.4  ...               2.3       2
149                5.9               3.0  ...               1.8       2

[150 rows x 5 columns]
```

Result:



D. LINK GOOGLE DRIVE

https://drive.google.com/file/d/1VPm5lHBTELHU70Qpe5KaKviBzascJHxB/view?usp=sharing

E. CHECKLIST

Poin	Ya	Tidak
1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan	√	
2. Convex hull yang dihasilkan sudah benar	√	
3. Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda.	√	
4. **Bonus**: program dapat menerima input dan menuliskan output untuk dataset lainnya.		√