LAPORAN TUGAS KECIL II

IF2211 STRATEGI ALGORITMA

Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer*

Disusun oleh:

Afnan Edsa Ramadhan 13521011

Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

BAB I

ALGORITMA DIVIDE AND CONQUER

A. Algoritma Divide and Conquer

Divide and Conquer awalnya adalah sebuah strategi militer yang dikenal dengan nama divide ut imperes. Sekarang strategi tersebut menjadi strategi fundamental di dalam ilmu komputer dengan nama *Divide and Conquer*. Untuk menyelesaikan algoritma *Divide and Conquer* dapat dibagi menjadi 3 langkah yaitu:

1. Divide
   Membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil.
2. Conquer
   Menyelesaikan masing-masing upa-persoalan secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar.
3. Combine
   Mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

B. Penerapan Algoritma

Penerapan Algoritma Divide and Conquer dalam mencari pasangan titik terdekat ini dapat dibagi menjadi beberapa tahapan dengan rincian sebagai berikut :

1. Masukkan semua titik ke dalam sebuah list
2. Mengurutkan semua titik berdasarkan salah satu sumbu, dalam algoritma ini penulis menggunakan sumbu X.
3. List lalu dibagi tepat ditengah menjadi upa list kiri dan upa list kanan
4. List akan dibagi terus secara rekursif hingga menyisakan 2 atau 3 titik dalam list tersebut
5. Jika tersisa dua titik maka akan langsung dihitung dengan rumus euclidian yang disediakan sedangkan jika tersisa tiga titik maka akan dihitung menggunakan algoritma brute force.
6. Dari langkah 5 akan didapatkan nilai-nilai terdekat dari masing-masing sub list, lalu nilai-nilai tersebut akan dibandingkan dengan semua nilai untuk mencari nilai terkecil.
7. Mencari titik-titik yang berada pada sekitar nilai tengah kemudian masukkan ke dalam list baru.
8. Dari langkah 7 cari nilai terkecil dari titik-titik yang berada di sekitar nilai tengah dengan menggunakan algoritma brute force.
9. Bandingakn kembali nilai yang didapat pada langkah 6 dengan nilai yang didapat pada langakh 8, untuk mencari mana nilai yang paling kecil.
10. Didapatkan jarak paling minimum dari seluruh titik-titik yang ada pada layar

BAB II

SOURCE CODE

fungsi.py

```
import random
import matplotlib.pyplot as plt
import numpy as np
import math
import sys

oppp = 0

def createDots(n,d):
    #n jumlah titik
    #d dimensi
    listt = []
    for i in range(n):
        temp = []
        for j in range(d):
            a = random.randint(0,99999)
            temp.append(a)
        listt.append(temp)
    return listt

def printList(list):
    for i in range(len(list)):
        for j in range(len(list[i])):
            print(list[i][j], end=" ")
        print("")

def show2dAll(list):
    for i in range(len(list)):
        x = list[i][0]
        y = list[i][1]
        plt.scatter(x, y, color='blue')

def showNearestDots2D(i,j):
    plt.scatter(i[0], i[1], color='red')
    plt.scatter(j[0], j[1], color='red')

def drawLine2D(i,j):
    print(i,j)
    x = np.array([i[0], j[0]])
    y = np.array([i[1], j[1]])
    plt.plot(x, y, color='red')

def show3dAll(list,ax):
    for i in range(len(list)):
        xs = list[i][0]
        ys = list[i][1]
```

```
        zs = list[i][2]
        ax.scatter(xs, ys, zs, color='blue')

def showNearestDots3D(i,j,ax):
    ax.scatter(i[0], i[1], i[2], color='red')
    ax.scatter(j[0], j[1], j[2], color='red')


def drawLine3D(i,j,ax):
    x = np.array([i[0], j[0]])
    y = np.array([i[1], j[1]])
    z = np.array([i[2], j[2]])
    ax.plot3D(x, y, z, color='red')

def hitungJarak(list,i,j):
    #i dan j indeks di list
    sum = 0
    for k in range(len(list[0])):
        sum+= (list[i][k]-list[j][k])**2
    jarak = math.sqrt(sum)
    return jarak

def findMinBruteForce(list):
    min = sys.maxsize
    for i in range(len(list)):
        for j in range(i+1,len(list)):
            # print(i,j,hitungJarak(list,i,j))
            temp = hitungJarak(list,i,j)
            if temp < min:
                min = temp
                titik1 = list[i]
                titik2 = list[j]
    return min,titik1,titik2

def splitList(list):
    n = len(list)//2
    if(n%2==0):
        return list[:n],list[n:]
    else:
        return list[:n],list[n+1:]

def findMinDivideConquer(list):
    #pembagian terkecil sisa 2 titik
    global oppp
    if(len(list)==2):
        oppp +=1
        return hitungJarak(list,0,1),list[0],list[1]
    elif(len(list)==3):
        oppp+=1
        return findMinBruteForce(list)
    else:
```

```
        l1,l2 = splitList(list)
        d1,t11,t12 = findMinDivideConquer(l1)
        d2,t21,t22 = findMinDivideConquer(l2)
        if(d1<d2):
            d = d1
            t1 = t11
            t2 = t12
        else:
            d = d2
            t1 = t21
            t2 = t22

        tengah = len(list)//2
        if(len(list)%2==0):
            avg = (list[tengah][0]+list[tengah+1][0])/2
        else:
            avg = list[tengah+1][0]
        #cek apakah ada titik yang ada di daerah -d dan +d
        #sekalian dimasukkin ke list temporary
        temp = []
        for i in list:
            if(i[0]>= avg-d and i[0]<= avg+d):
                temp.append(i)

        #cari minimum jarak yang ada di daerah -d dan +d
        for i in range(len(temp)):
            for j in range(i+1,len(temp)):
                for k in range(len(temp[0])):
                    if(abs(temp[i][k]-temp[j][k])>d):
                        flag = False
                        break
                    else:
                        flag = True
                if(flag):
                    d3 = hitungJarak(temp,i,j)
                    oppp +=1
                    if(d3<d):
                        d = d3
                        t1 = temp[i]
                        t2 = temp[j]
        return d,t1,t2

def showVisual(dim,list,titik1,titik2,distance):
    #d dimensi
    #titik1 dan titik2 indeks di list
    if(dim==2):
        print("2D")
        show2dAll(list)
        showNearestDots2D(titik1,titik2)
        drawLine2D(titik1,titik2)
```

```
        plt.annotate(round(distance), xy=((titik1[0]+titik2[0])/2,
(titik1[1]+titik2[1])/2))
        plt.xlabel("Sumbu X")
        plt.ylabel("Sumbu Y")
        plt.show()

    elif(dim==3):
        print("3D")
        ax = plt.axes(projection='3d')
        show3dAll(list,ax)
        showNearestDots3D(titik1,titik2,ax)
        drawLine3D(titik1,titik2,ax)
        ax.set_xlabel('Sumbu X')
        ax.set_ylabel('Sumbu Y')
        ax.set_zlabel('Sumbu Z')
        plt.show()


def showASCIIART():
    print("       ___          _      __                ___      ")
    print("  ____/ (_)    __(_)___/ /__      ( _ )")
    print(" / __  / / | / / / __  / _ \    / __ \/|")
    print("/ /_/ / /| |/ / / /_/ /  __/  / /_/  <")
    print("\__,_/_/ |___/_/\__,_/\___/   \____/\/")
    print("  __________  ____  ____ ___  _____  _____")
    print(" / ___/ __ \/ __ \/ __ `/ / / / _ \/ ___/")
    print("/ /__/ /_/ / / / / /_/ / /_/ /  __/ /")
    print("\___/\____/_/ /_/\__, /\__,_/\___/_/")
    print("                  /_/")


def countOperation():
    return oppp
```

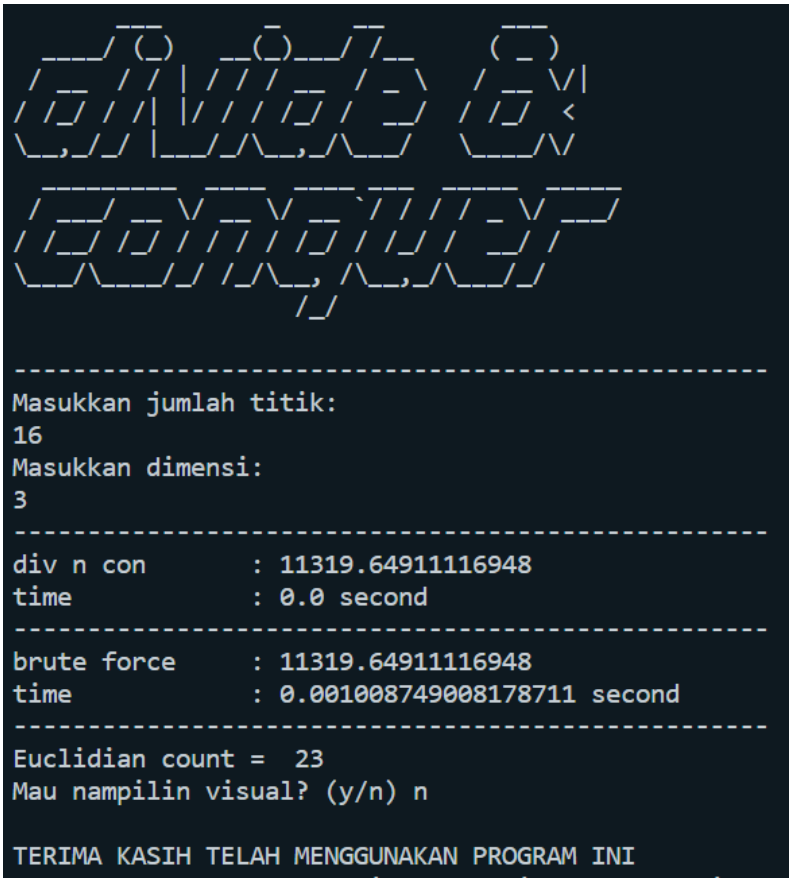
main.py

```
from fungsi import *
import time

def main():
    showASCIIART()
    print("\n-----------------------------------------------------")
    print("Masukkan jumlah titik: ")
    n = int(input())
    print("Masukkan dimensi: ")
    d = int(input())
    myList = createDots(n,d)
    myList.sort()

    startd = time.time()
    dnc,t1,t2 = findMinDivideConquer(myList)
```

```
    endd = time.time()

    print("-----------------------------------------------------")
    print("div n con \t:", dnc)
    print("time\t\t:", endd-startd, "second")

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

    startb = time.time()
    bf = findMinBruteForce(myList)[0]
    endb = time.time()
    print("brute force \t:",bf)
    print("time\t\t:", endb-startb, "second")
    print("-----------------------------------------------------")

    print("Euclidian count = ", countOperation())

    vis = input("Mau nampilin visual? (y/n) ")
    if(vis == 'y' or vis == 'Y'):
        if(d<=3):
            if(n>=1000):
                vis = input("Yakin mau nampilin visual?(ngelag
banget ntar) (y/n) ")
                if(vis == 'y' or vis == 'Y'):
                    print("SIAPPPPP")
                    print("Resiko ditanggung sendiri")
                    showVisual(d,myList,t1,t2,dnc)
            else:
                showVisual(d,myList,t1,t2,dnc)
        else:
            print("MAAF VISUALISASI TIDAK DAPAT DITAMPILKAN")
            print("KARENA DIMENSI MELEBIHI 3")

    print("\nTERIMA KASIH TELAH MENGGUNAKAN PROGRAM INI")


if __name__ == "__main__":
    main()
```

BAB III
TEST CASE

A. N = 16



```
 ___       _     __        ___
  ____/ (_)   __(_)___/ /__     ( _ )
 / __  / / | / / / __  / _ \   / __ \/|
/ /_/ / /| |/ / / /_/ /  __/  / /_/  <
\__,_/_/ |___/_/\__,_/\___/   \____/\/
  _________  ____  ____ ___  _____  _____
 / ___/ __ \/ __ \/ __ `/ / / / _ \/ ___/
/ /__/ /_/ / / / / /_/ / /_/ /  __/ /
\___/\____/_/ /_/\__, /\__,_/\___/_/
                /_/

----------------------------------------------------
Masukkan jumlah titik:
16
Masukkan dimensi:
3
----------------------------------------------------
div n con       : 11319.64911116948
time            : 0.0 second
----------------------------------------------------
brute force     : 11319.64911116948
time            : 0.001008749008178711 second
----------------------------------------------------
Euclidian count =  23
Mau nampilin visual? (y/n) n

TERIMA KASIH TELAH MENGGUNAKAN PROGRAM INI
```

B. N = 64

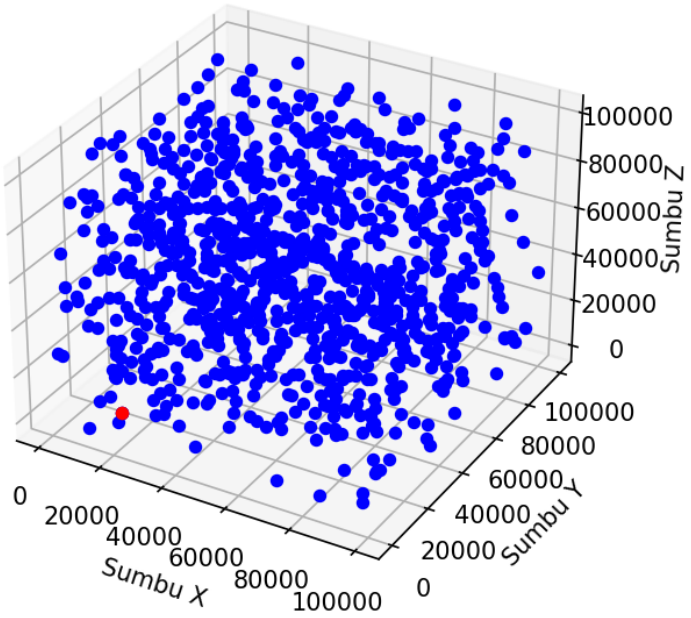
```
----------------------------------------------------
Masukkan jumlah titik:
64
Masukkan dimensi:
3
----------------------------------------------------
div n con       : 4057.712409720531
time            : 0.0019958019256591797 second
----------------------------------------------------
brute force     : 4057.712409720531
time            : 0.007996797561645508 second
----------------------------------------------------
Euclidian count =  77
```

C. N = 128

```
-----------------------------------------------------
Masukkan jumlah titik:
128
Masukkan dimensi:
3
-----------------------------------------------------
div n con       : 2660.0543227535786
time            : 0.0029904842376708984 second
-----------------------------------------------------
brute force     : 2660.0543227535786
time            : 0.024996042251586914 second
-----------------------------------------------------
Euclidian count =  161
Mau nampilin visual? (y/n) n
```

D. N = 1000

```
-----------------------------------------------------
Masukkan jumlah titik:
1000
Masukkan dimensi:
3
-----------------------------------------------------
div n con       : 919.1746297630282
time            : 0.04703927040100098 second
-----------------------------------------------------
brute force     : 919.1746297630282
time            : 0.746800422668457 second
-----------------------------------------------------
Euclidian count =  715
Mau nampilin visual? (y/n) n
```



E. Spesifikasi Laptop

Pengujian dilakukan menggunalan laptop dengan spesifikasi sebagai berikut :

a. Processor : Intel Core i7-8550U
b. Memory : 8 Gb
c. OS : Windows 10

LAMPIRAN

Link Git-Hub : https://github.com/afnanramadhan/Tucil2_13521011

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil running	✓	
3. Program dapat menerima masukan dan dan menuliskan luaran.	✓	
4. Luaran program sudah benar (solusi closest pair benar)	✓	
5. Bonus 1 dikerjakan	✓	
6. Bonus 2 dikerjakan	✓	