IF2211 - Strategi Algoritma
Laporan Tugas Kecil 2

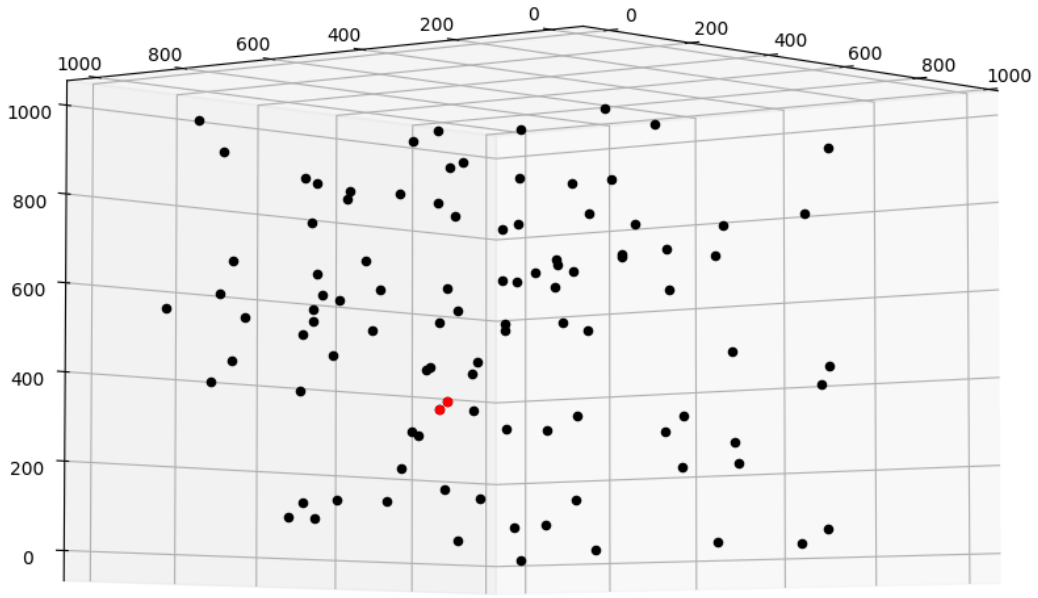
Disusun Oleh:

Henry Anand Septian Radityo	**13521004**
Haikal Ardzi Shofiyyurrohman	**13521012**

Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Tahun Ajaran 2022/2023

Daftar Isi

1 Deskripsi Masalah



1.1 Deskripsi Umum Persoalan

Mencari pasangan titik terdekat dengan Algoritma Divide and Conquer sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tugas kecil 2 kali ini kami diminta mengembangkan algoritma mencari pasangan titik terdekat pada bidang 3D. Misalkan terdapat n buah titik pada ruang 3D. Setiap titik P di dalam ruang dinyatakan dengan koordinat P = (x, y, z). Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titik P1 = (x1, y1, z1) dan P2 = (x2, y2, z2) dihitung dengan rumus *Euclidean* berikut:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Lalu program ini akan menerapkan Algoritma *divide and Conquer* dalam mencari pasangan titik dengan jarak terdekat tersebut dan membandingkan dengan Algoritma *Brute Force*.

2 Metode Penyelesaian

2.1 Definisi Umum Algoritma Divide and Conquer

Algoritma divide and conquer adalah algoritma yang digunakan untuk memecahkan persoalan dengan cara membagi sebuah persoalan menjadi beberapa upa persoalan. Pembagian persoalan menjadi beberapa bagian tersebut diharapkan akan membantu mereduksi kompleksitas dari masalah sehingga lebih mudah untuk diselesaikan.

Setelah persoalan dibagi - bagi, maka proses Divide and Conquer akan melakukan proses Conquer atau proses penyelesaian. Pada proses ini, upa - upa persoalan akan diselesaikan satu per satu. Apabila semua proses sudah diselesaikan maka algoritma akan menggabungkan semua solusi dari upa persoalan untuk mendapatkan solusi penuh dari persoalan.

Algoritma Divide and Conquer memiliki karakteristik yang sama pada setiap upa persoalan. Hal ini menyebabkan algoritma ini sering diselesaikan dengan menggunakan skema pemrograman rekursif.

2.2 Penerapan Algoritma Divide and Conquer Pada Pencarian Pasangan Titik Terdekat 3D

Langkah-langkah menggunakan Algoritma *Divide and Conquer* dalam mencari pasangan titik terdekat ini dapat dibagi menjadi serangkaian tahap. Berikut tahapan dari Algoritma tersebut:

1. Memasukkan seluruh titik (*point)* yang ada ke dalam *List of Point*.

2. Mengurutkan seluruh elemen berdasarkan nilai elemen titik X dengan terurut membesar pada *List of Point*

3. Secara rekursif membagi 2 *List of Point* secara rata dengan referensi rata-rata elemen titik X (***averageX***) pada setiap pembagian dari seluruh titik, lalu didapatkan 2 *List of Point* sementara (***tempListOfPoint)*** dengan ***tempListofPoint*** 1 < ***averageX*** dan ***tempListofPoint*** *2* > ***averageX***, lalu dibagi kembali dengan metode yang sama secara terus-menerus hingga hanya tersisa 2 atau 3 elemen yang tersisa.

4. Jika 2 maka langsung menghitung nilai jarak dengan rumus *euclidean*, jika 3 maka menghitung nilai jarak untuk setiap pasangan titik, lalu melakukan brute force untuk mendapatkan pasangan titik terdekat dan mengembalikan nilai jarak tersebut sebagai nilai jarak **minimum** pada ***tempListOfPoint*.**

5. Membandingkan kedua jarak yang didapat pada kedua ***tempListOfPoint***, sehingga mendapatkan nilai **minimum** dari perbandingan tersebut.

6. Lalu melakukan pengecekan jarak **minimum** pasangan titik pada sekitar wilayah ***averageX*+minimum** dan ***averageX*-minimum** juga yang memungkinkan memiliki jarak lebih pendek daripada **minimum**, jika didapatkan maka jarak tersebut akan menjadi nilai **minimum**.

7. Didapatkan jarak **minimum** pasangan titik terdekat pada seluruh titik tersebut.

3 Eksperimen

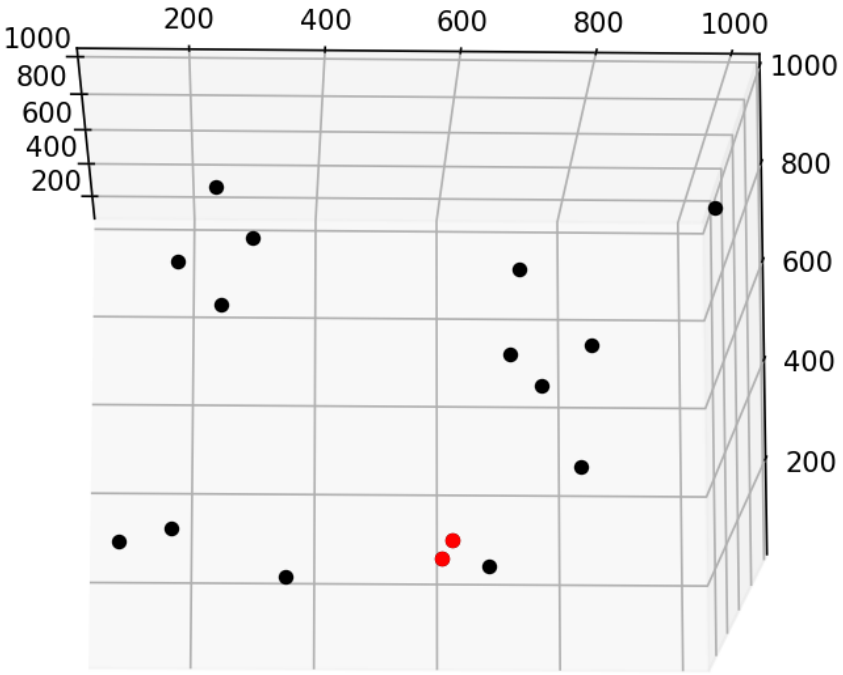
3.1 Kasus numberOfPoint=16

```
Masukkan banyak Titik (n): 16
Masukkan Dimensi: 3

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 68.03 points
Execution Time: 0.99 ms
Euclidean Operation in DnC Count: 168
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 68.03 points
Execution Time: 1.20 ms
Euclidean Operation in BF Count: 243
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.1.1 Kasus n = 16 dan Dimensi = 3



Gambar 3.1.2 Kasus n = 16 dan Dimensi = 3

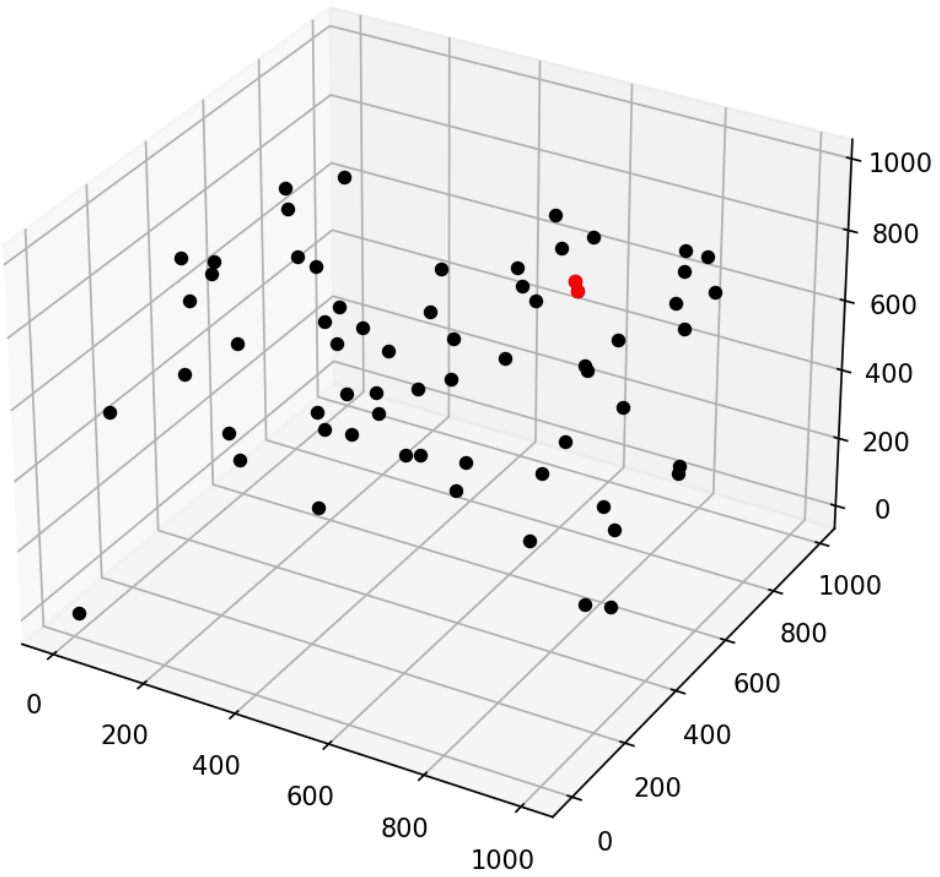
3.2 Kasus numberOfPoint=64

```
Masukkan banyak Titik (n): 64
Masukkan Dimensi: 3

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 51.42 points
Execution Time: 6.48 ms
Euclidean Operation in DnC Count: 2312
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 51.42 points
Execution Time: 7.83 ms
Euclidean Operation in BF Count: 4040
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.2.1 Kasus n = 64 dan Dimensi = 3



Gambar 3.2.2 Kasus n = 32 dan Dimensi = 3

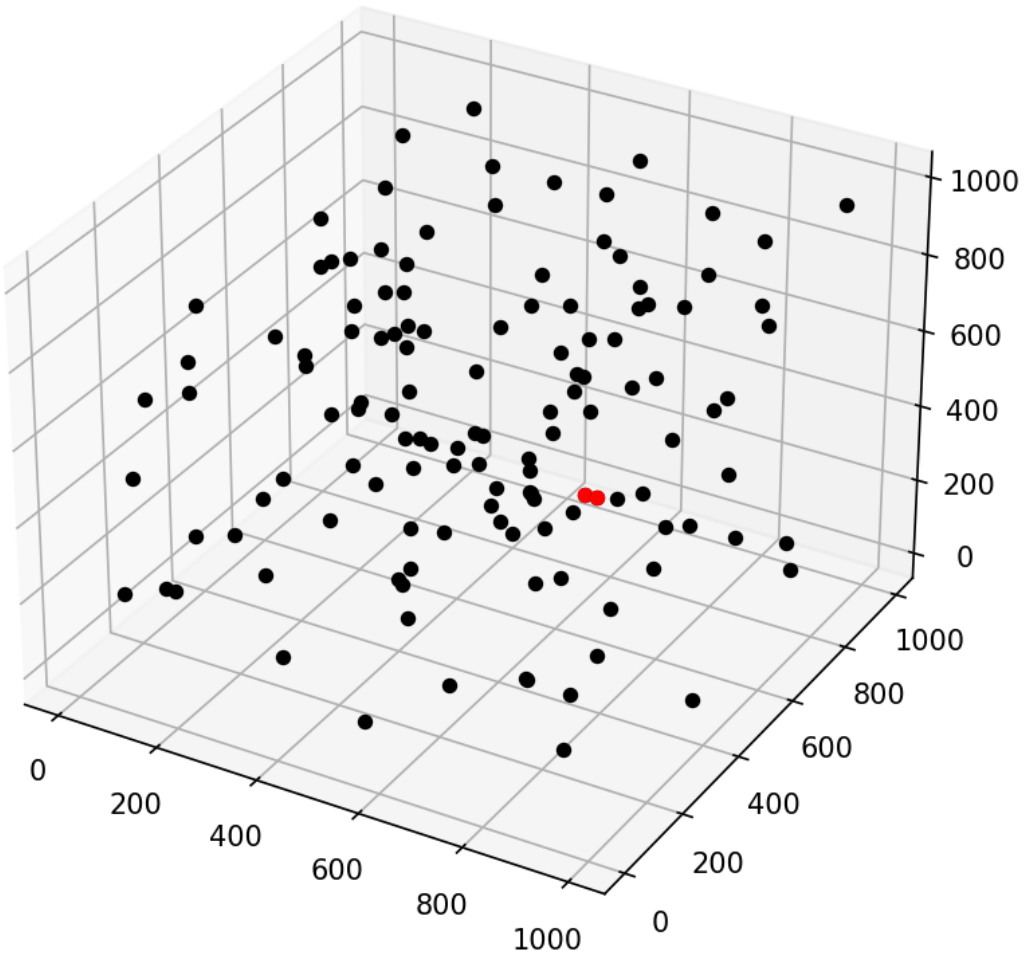
3.3 Kasus numberOfPoint=128

```
Masukkan banyak Titik (n): 128
Masukkan Dimensi: 3

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 28.04 points
Execution Time: 13.61 ms
Euclidean Operation in DnC Count: 9428
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

==============[BRUTE FORCE]==============
Distance with BF Algorithm : 28.04 points
Execution Time: 21.60 ms
Euclidean Operation in BF Count: 16267
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.3.1 Kasus n = 128 dan Dimensi = 3



Gambar 3.3.2 Kasus n = 128 dan Dimensi = 3

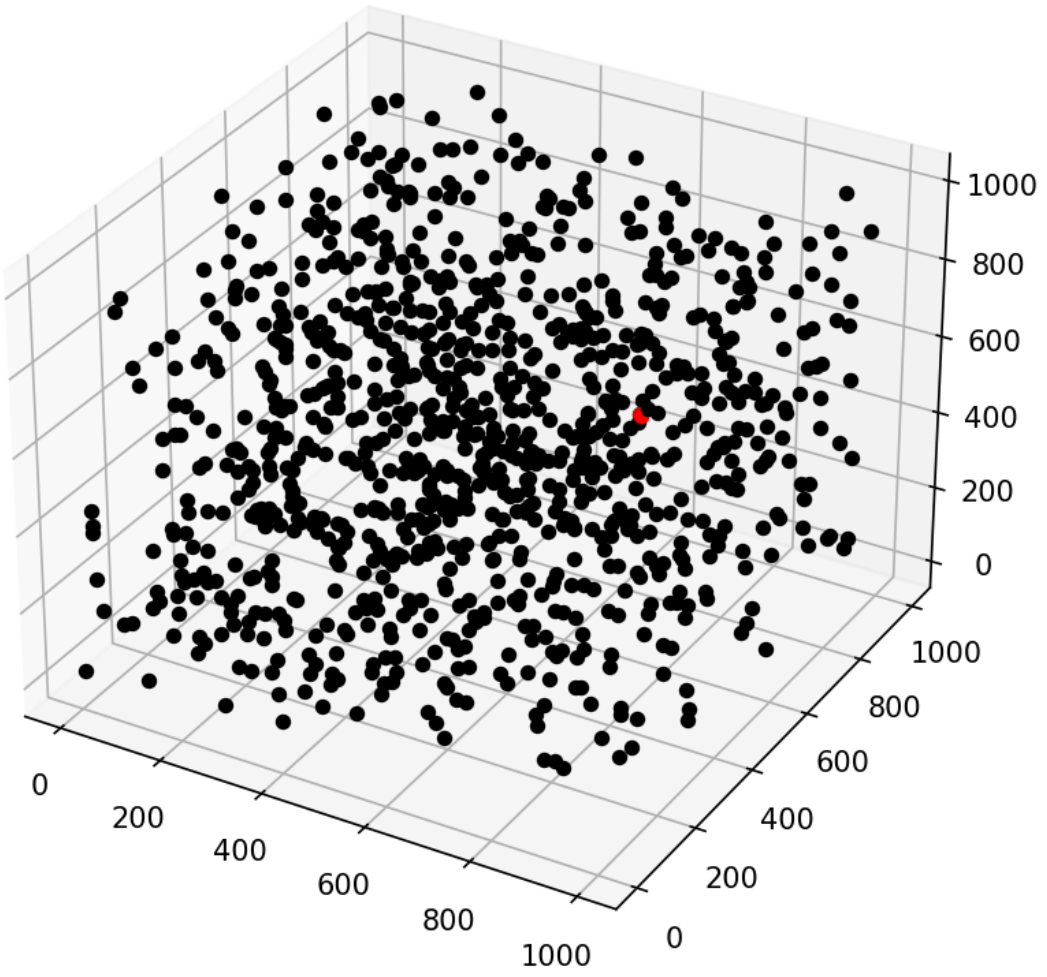
3.4 Kasus numberOfPoint=1000

```
Masukkan banyak Titik (n): 1000
Masukkan Dimensi: 3

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 7.30 points
Execution Time: 239.90 ms
Euclidean Operation in DnC Count: 172474
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 7.30 points
Execution Time: 1378.46 ms
Euclidean Operation in BF Count: 999019
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.4.1 Kasus n = 1000 dan Dimensi = 3



Gambar 3.4.2 Kasus n = 1000 dan Dimensi = 3

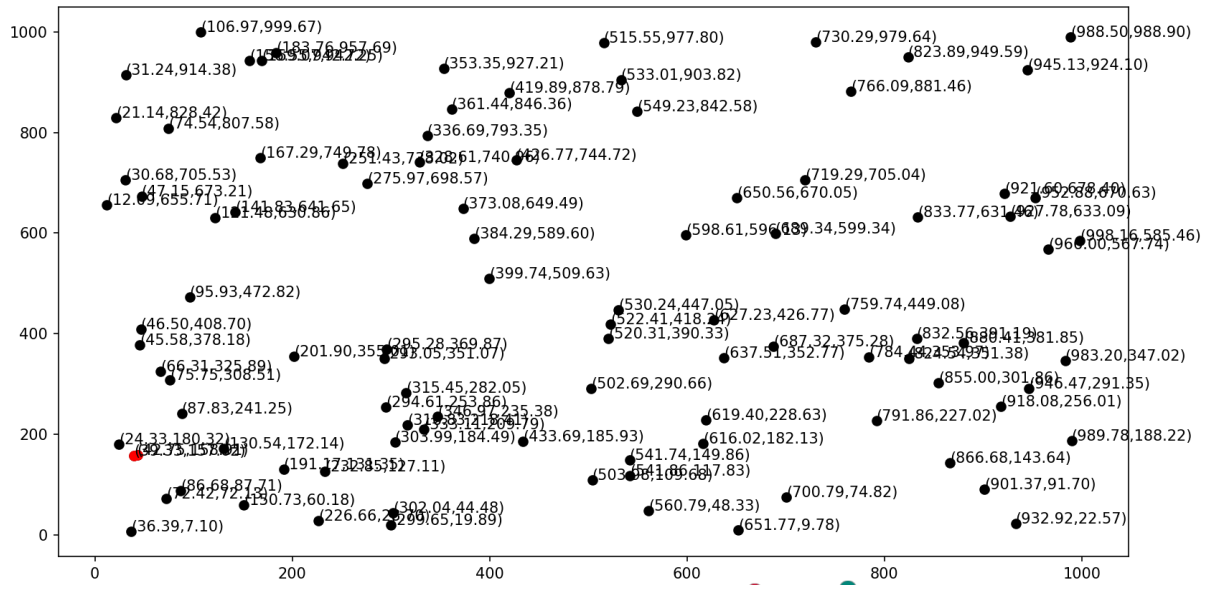
3.5 Kasus dimensi != 3

```
Masukkan banyak Titik (n): 100
Masukkan Dimensi: 2

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 3.65 points
Execution Time: 4.22 ms
Euclidean Operation in DnC Count: 2298
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 3.65 points
Execution Time: 12.06 ms
Euclidean Operation in BF Count: 9905
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.5.1 Kasus n = 100 dan Dimensi = 2



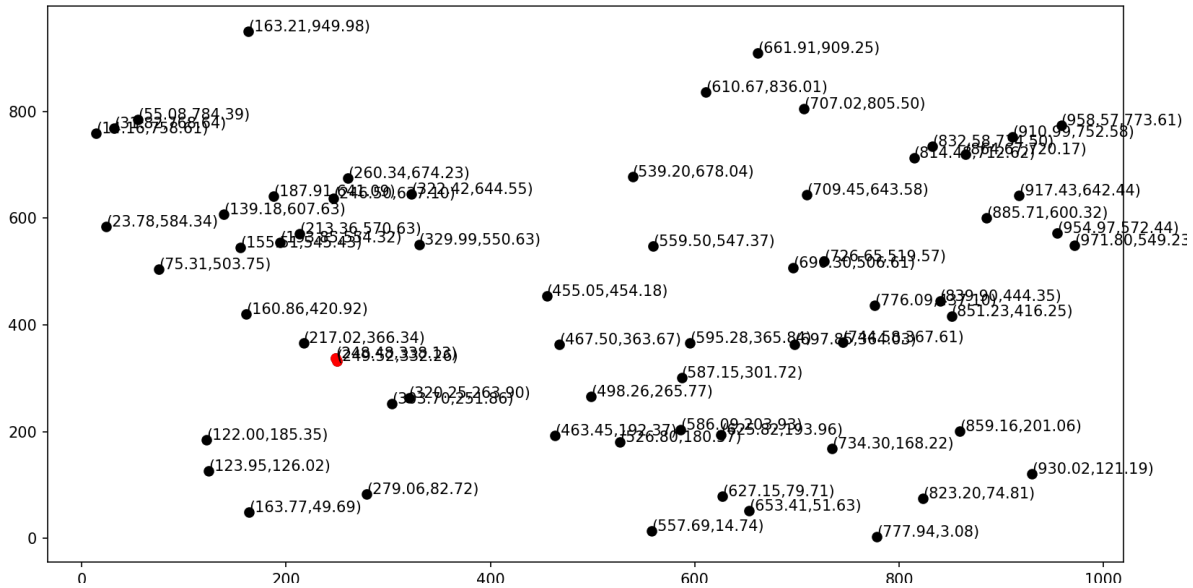
Gambar 3.5.2 Kasus n = 100 dan Dimensi = 2

```
Masukkan banyak Titik (n): 64
Masukkan Dimensi: 2

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 5.97 points
Execution Time: 5.05 ms
Euclidean Operation in DnC Count: 1104
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 5.97 points
Execution Time: 8.52 ms
Euclidean Operation in BF Count: 4035
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.5.3 Kasus n = 64 dan Dimensi = 2



Gambar 3.5.4 Kasus n = 64 dan Dimensi = 2

```
Masukkan banyak Titik (n): 100
Masukkan Dimensi: 4

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 25.95 points
Execution Time: 17.55 ms
Euclidean Operation in DnC Count: 8710
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 25.95 points
Execution Time: 21.14 ms
Euclidean Operation in BF Count: 9915
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.5.5 Kasus n = 100 dan Dimensi = 4

```
Masukkan banyak Titik (n): 100
Masukkan Dimensi: 5

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 132.39 points
Execution Time: 28.55 ms
Euclidean Operation in DnC Count: 13686
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 132.39 points
Execution Time: 19.06 ms
Euclidean Operation in BF Count: 9906
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.5.6 Kasus n = 100 dan Dimensi = 5

```
Masukkan banyak Titik (n): 1000
Masukkan Dimensi: 6

============[DIVIDE AND CONQUER]============
Distance With DnC Algorithm 83.67 points
Execution Time: 1659.06 ms
Euclidean Operation in DnC Count: 956258
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor

===============[BRUTE FORCE]===============
Distance with BF Algorithm : 83.67 points
Execution Time: 1815.72 ms
Euclidean Operation in BF Count: 999008
Run in Intel64 Family 6 Model 126 Stepping 5, GenuineIntel processor
```

Gambar 3.5.7 Kasus n = 1000 dan Dimensi = 6

4 Source Code (Python)

4.1. main.py

```
import Point as src
import random
import time
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import platform

# ASCII ART
# Dihapus untuk menghemat tempat

# input banyaknya titik
numberOfPoints = int(input("Masukkan banyak Titik (n): "))

# input dimensi
dimension = int(input("Masukkan Dimensi: "))

# random int setiap elemen titik
arrayPoint = [[0 for j in range(dimension)] for i in range (numberOfPoints)]
for i in range(numberOfPoints):
    for j in range(dimension):
        arrayPoint[i][j] = random.uniform(0, 1000)

# sorting arrayPoint berdasarkan X membesar
arrayPoint=src.sortArrOfPoint(arrayPoint)
startTime1 = time.time()
result = src.findClosestPairDnC(arrayPoint, numberOfPoints, dimension)
print()
print("============[DIVIDE AND CONQUER]============")
print("Distance With DnC Algorithm "+
"{:.2f}".format(round((src.findClosestPairDnC(arrayPoint, numberOfPoints,
dimension)[0]),2)) + " points")
resTimeDnC=(time.time()-startTime1)*1000
print("Execution Time: " + "{:.2f}".format(round((resTimeDnC),2))+" ms")
eucCountDnC = src.eucCount
print("Euclidean Operation in DnC Count: "+str(eucCountDnC))
print("Run in " + str(platform.processor()) + " processor")
print()
print("===============[BRUTE FORCE]===============")
startTime2 = time.time()
print("Distance with BF Algorithm : "+
"{:.2f}".format(round((src.findClosestPairBruteforce(arrayPoint)[0]),2))+ "
points")
resTimeBF=(time.time()-startTime2)*1000
print("Execution Time: " + "{:.2f}".format(round((resTimeBF),2))+" ms")
eucCountBF = src.eucCount-eucCountDnC
print("Euclidean Operation in BF Count: "+str(eucCountBF))
```

```
print("Run in " + str(platform.processor()) + " processor")
print("")

# visualisasi
fig = plt.figure(figsize=(100,100))

# dimensi 3
if dimension==3:
    tempdim = str(dimension)+"d"
    ax = fig.add_subplot(111, projection=tempdim)
    for i in range(len(arrayPoint)):

ax.scatter(arrayPoint[i][0],arrayPoint[i][1],arrayPoint[i][2],c='black',marker=
'o')
    for i in range(1,len(result)):
        ax.scatter(result[i][0],result[i][1],result[i][2],c='red',marker='o')
    plt.show()
# dimensi 2
elif dimension==2:
    for i in range(len(arrayPoint)):
        plt.scatter(arrayPoint[i][0],arrayPoint[i][1],color = 'black')
    for i in range(1,len(result)):
        plt.scatter(result[i][0],result[i][1],color = 'red')
    x = []
    y = []
    for i in range(len(arrayPoint)):
        x.append(arrayPoint[i][0])
        y.append(arrayPoint[i][1])
    for xy in zip(x,y):
        plt.annotate('(%.2f,%.2f)'% xy, xy = xy)
    plt.show()
```

4.2. Point.py

```
import math
import sys
eucCount = 0
# calculateDistance between 2 point using euclidean
def calculateDistance(point1, point2):
    global eucCount
    eucCount+=1
    result=float(0)
    for i in range (len(point1)):
        result+=(point2[i]-point1[i])**2
    return math.sqrt(result)
# print point in rows
def printAllPoint(arrayPoint):
    for i in range(len(arrayPoint)):
        print(arrayPoint[i])
```

```
# find closest pair in brute force algorithm
def findClosestPairBruteforce(arrayPoint):
    temp=float(calculateDistance(arrayPoint[0], arrayPoint[1]))
    ret = [temp,arrayPoint[0],arrayPoint[1]]
    for i in range(len(arrayPoint)):
        for j in range(len(arrayPoint)):
            if i!=j:
                if temp>calculateDistance(arrayPoint[i], arrayPoint[j]):
                    temp=calculateDistance(arrayPoint[i], arrayPoint[j])
                    ret = [temp, arrayPoint[i],arrayPoint[j]]
    return ret

# find closest pair in divide and conquer algorithm
def findClosestPairDnC(arr, n, dimensi):
    if (n==3):
        d1 = calculateDistance(arr[0],arr[1])
        d2 = calculateDistance(arr[0],arr[2])
        d3 = calculateDistance(arr[1],arr[2])
        if (d1>=d2 and d1>=d3):
            ret = []
            ret.append(d1)
            ret.append(arr[0])
            ret.append(arr[1])
            return ret
        elif (d2>=d1 and d2>=d3):
            ret = []
            ret.append(d2)
            ret.append(arr[0])
            ret.append(arr[2])
            return ret
        else:
            ret = []
            ret.append(d3)
            ret.append(arr[1])
            ret.append(arr[2])
            return ret
    elif (n==2):
        d = calculateDistance(arr[0],arr[1])
        ret = []
        ret.append(d)
        ret.append(arr[0])
        ret.append(arr[1])
        return ret
    else:
        mid = n//2
        arr1 = arr[:mid]
        arr2 = arr[mid:]

        d1 = findClosestPairDnC(arr1,mid,dimensi)
        d2 = findClosestPairDnC(arr2,mid,dimensi)
```

```
        d = min(d1[0],d2[0])
        if(d1[0]<d2[0]):
            ret = d1
        else:
            ret = d2
        tempResult=[]

        if(n%2==0):
            avg=(arr[n//2][0]+arr[n//2+1][0])/2
        else:
            avg=arr[n//2][0]

        for i in range(len(arr)):
            if arr[i][0]<=avg+d and arr[i][0]>=avg-d:
                tempResult.append(arr[i])

        if (len(tempResult)>=2):
            temp=float(calculateDistance(tempResult[0],tempResult[1]))
            ti = 0
            tj = 1
            for i in range (len(tempResult)):
                for j in range(len(tempResult)):
                    if i!=j:
                        if temp>calculateDistance(tempResult[i],tempResult[j]):
                            temp=calculateDistance(tempResult[i],tempResult[j])
                            ti = i
                            tj = j
        if (len(tempResult)>=2):
            if(temp<ret[0]):
                ret = [temp,tempResult[ti],tempResult[tj]]
                return ret
            else:
                return ret
        else:
            return ret

# sorting point using X1 value ascending
def sortArrOfPoint(arr):
    for i in range(len(arr)):
        tempx = arr[i][0]
        for j in range(i,len(arr)):
            if (tempx>=arr[j][0]):
                tempx = arr[j][0]
                idxfound = j
        temp = arr[i]
        arr[i] = arr[idxfound]
        arr[idxfound] = temp
    return arr
```

5 Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada 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	☑	

6 Repository Github

Open https://github.com/henryanandsr/Tucil2_13521004_13521012.git