LAPORAN TUGAS KECIL 2

IF2211 Strategi Algoritma

Find Closest Pairs

Menentukan Titik Terdekat dengan Algoritma *Divide and Conquer*

Oleh:

Agsha Athalla Nurkareem

13521027

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

DAFTAR ISI

BAB I

Algoritma Divide and Conquer

1.1. Pengertian Algoritma Divide and Conquer

Algoritma Divide and Conquer berasal dari kata divide yang artinya membagi persoalan menjadi lebih kecil dan conquer yang berarti menyelesaikan masing-masing upa-persoalan. Sehingga Algoritma divide and Conquer memiliki makna bahwa persoalan dapat dibuat menjadi beberapa upa-persoalan yang memiliki persoalan dengan kemiripan yang sama namun berukuran lebih kecil. Persoalan dengan ukuran yang lebih kecil itulah yang nantinya akan diselesaikan.

Dalam algoritma Divide and Conquer memiliki tiga proses, yaitu divide, conquer, dan combine. Pertama kali algoritma akan membagi permasalahan ke dalam ukuran yang lebih kecil. Selanjutnya, setiap upa-permasalahan tersebut akan diselesaikan secara rekursif dengan tujuan untuk mencari solusi dari setiap upa-permasalahan tersebut. Setelahnya, upa-permasalahan tersebut digabung untuk menyelesaikan masalah utama. Algoritma ini terbukti lebih efektif dibandingkan dengan algoritba Brute Force karena kecepatannya dalam menemukan jawaban. Tetapi algoritma ini memiliki kekurangan yaitu terkadang hanya dapat menunjukkan solusi optimum lokal dibandingkan dengan algoritma Brute Force yang selalu memiliki solusi optimum global.

1.2. Penggunaan Algoritma Divide and Conquer dalam program

Untuk mendapatkan hasil pasangan titik terdekat, penulis menggunakan algoritma Divide and Conquer. Pertama-tama, penulis menyusun banyaknya titik dalam array of array. Kemudian array tersebut akan diurutkan berdasarkan titik absis (sumbu x). Setelah array tersusun berdasarkan nilai absis terkecil hingga terbesar, bagi array tersebut di tengah. Apabila titik berjumlah 2, maka program akan langsung selesai. Setelah membagi 2 array (array kiri dan array kanan), apabila jumlah matriks setelah dibagi menjadi 2, maka titik

terdekat langsung ditemukan. Apabila jumlah matriks setelah dibagi menjadi 3, maka cari 2 titik terdekat dengan menggunakan algoritma Brute Force. Ketika jumlah matriks setelah dibagi lebih dari 3, matriks akan dibagi lagi secara rekursif sampai ukuran array menjadi 2 atau 3 dan diselesaikan dengan kedua cara sebelumnya. Terdapat kondisi ketika membagi suatu titik di mana titik terdekat berada di antara pembagi wilayah. Untuk menyelesaikan jarak terdekat dari 2 titik, dapat dicari dengan rumus Euclidean berikut.

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

BAB II

Source Code dalam Bahasa Python

2.1 File splashScreen.py

```
def splashScreen():
    print('''


            ______ _           _    _____
_                    _     _____      _
            |  ____(_)        | |  / ____| |                  | |   |  __
\    (_)
            | |__   _ _ __   __| | | |    | | ___  ___  ___  ___| |_  | |__)
|_ _ _ _ __
            |  __| | | '_ \ / _` | | |    | |/ _ \/ __|/ _ \/ __| __| |  ___/
_` | | '__|
            | |    | | | | | (_| | | |____| | (_) \__ \  __/\__ \ |_  | |  |
(_| | | |
            |_|    |_|_| |_|\__,_|  \_____|_|\___/|___/\___||___/\__|
|_|   \__,_|_|_|


                                ------------------------------
                                Nama : Agsha Athalla Nurkareem
                                NIM : 13521027
                                Mata Kuliah : IF2211
                                ------------------------------


    ''')
```

2.2 File function.py

```
import random
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import bruteForce as t

def inputan():
    n = int(input("Masukkan banyaknya titik: "))
    while (n<=1):
        print("Banyaknya titik tidak boleh kurang dari 2!")
        n = int(input("Masukkan banyaknya titik: "))
```

```
    d = int(input("Masukkan banyaknya dimensi: "))
    while (d<1):
        print("Banyaknya dimensi tidak boleh kurang dari 1!")
        d = int(input("Masukkan banyaknya dimensi: "))

    matriks = [[float(0) for j in range(d)]for i in range(n)]
    for i in range(n):
        for j in range(d):
            matriks[i][j] = round(random.uniform(-100, 100),3)
    return n, d, matriks

def buatPembanding(array, pertama, kedua, d):
    arrayPembanding = [[0 for j in range(d)]for i in range(2)]
    for j in range(d):
        arrayPembanding[0][j] = array[pertama][j]
        arrayPembanding[1][j] = array[kedua][j]
    return arrayPembanding

def nilaiEuclidean(array,d,euclideanCount):
    sum = 0
    arrayPembanding = array
    for j in range(d):
        sum += (array[0][j] - array[1][j])**2
    sum = math.sqrt(sum)
    euclideanCount += 1
    return sum, arrayPembanding, euclideanCount

def nilaiEuclidean3(array,d, count):
    shortest = 99999999
    titik = [[0 for i in range(d)] for i in range(2)]
    for i in range(3):
        for j in range(3-i):
            if (i!=j):
                bandingkan = buatPembanding(array, i, j, d)
                eucledian, a, count = nilaiEuclidean(bandingkan, d, count)
                if (eucledian < shortest):
                    shortest = eucledian
                    titik = a
    return shortest, titik

def sortByX(array, n):
    for i in range(n):
        for j in range(n-i-1):
            if array[j][0] > array[j+1][0]:
                array[j], array[j+1] = array[j+1], array[j]
    return array

def minimum(a,b):
```

```
    if (a<b):
        return a
    else:
        return b

def divide(array, n, d):
    M1 = []
    M2 = []
    for i in range(n):
        if (i < n//2):
            M1.append(array[i])
        else:
            M2.append(array[i])
    return M1, M2
```

2.3 File showGraph.py

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def visualisasi(titikAsal, titikAkhir, n, d):
    fig = plt.figure(figsize=(12,15))
    if(n>2):
        titikAsal.remove(titikAkhir[0])
        titikAsal.remove(titikAkhir[1])

    titikAsal = np.array(titikAsal)
    titikAkhir = np.array(titikAkhir)

    if(d==3):
        ax = plt.axes(projection='3d')

        x = titikAsal[:,0]
        y = titikAsal[:,1]
        z = titikAsal[:,2]
        x1 = titikAkhir[:,0]
        y1 = titikAkhir[:,1]
        z1 = titikAkhir[:,2]

        ax.scatter(x,y,z, s=10, color='grey')
        ax.scatter(x1, y1, z1, s=20, color='red')

        plt.show()

    elif (d==2):
```

```
        x = titikAsal[:,0]
        y = titikAsal[:,1]
        x1 = titikAkhir[:,0]
        y1 = titikAkhir[:,1]

        plt.scatter(x,y, s = 10, color='grey')
        plt.scatter(x1, y1, s = 20, color='red')

        plt.show()
```

2.4 File main.py

```
import function as f
import divideAndConquer as dnc
import bruteForce as bf
import time
import showGraph as sg
import splashScreen as ss
import os

def main():
    os.system('cls')
    ss.splashScreen()
    euclideanCount = 0
    n, d, matriks = f.inputan()
    matriks = f.sortByX(matriks, n)
    detik1 = time.time()
    jarak, titik, count = dnc.FindClosestPair(matriks, n, d, euclideanCount)
    print('''
_____________________________________________________________________________

                        ALGORITMA DIVIDE AND CONQUER
    ''')
    print("\n")
    print("jarak terdekat = ",jarak)
    print("titik = ",titik)
    detik2 = time.time()
    print("execute time = ",detik2-detik1, " seconds")
    print("euclidean count = ",count)
    print("\n")
    print("_____________________________________________________________________________
_____")
    print('''
                            ALGORITMA BRUTE FORCE
    ''')

    detikBF1 = time.time()
    jarak1, titik1, count = bf.bruteForce(matriks, n, d, 0)
```

```
    print("jarak brute force = ",jarak1)
    print("titik brute force = ",titik1)
    detikBF2 = time.time()
    print("execute time brute force = ",detikBF2-detikBF1, " seconds")
    print("euclidean count = ",count)
    print("\n")
    print("_____________________________________________________________________
_____")

    print('''
                                VISUALISASI
    ''')

    if (d==2 or d==3):
        y = str(input("Apakah anda ingin memvisualisasikan hasil pengerjaan?
(y/n) "))
        while (y!='y' and y!='n'):
            print("masukan salah")
            y = str(input("Apakah anda ingin memvisualisasikan hasil
pengerjaan? (y/n) "))
        if (y == 'y'):
            print("Visualisasi akan ditampilkan. Titik merah menunjukkan
pasangan titik terdekat")
            print("Harap menutup layar visualisasi setelah selesai untuk
melanjutkan program")
            sg.visualisasi(matriks, titik, n, d)

    else:
        print("Anda tidak dapat memvisualisasikan hasil pengerjaan karena
bukan dalam dimensi 2 (dua) atau 3 (tiga)")

    print("\n")
    print("_____________________________________________________________________
_____")

    y = str(input("Apakah anda ingin mencoba kembali? (y/n) "))
    while (y!='y' and y!='n'):
        print("masukan salah")
        y = str(input("Apakah anda ingin mencoba kembali? (y/n) "))
    if (y == 'y'):
        main()
    elif (y == 'n'):
        os.system('cls')
        ss.splashScreen()
        print("                          🙏 Terima kasih telah menggunakan
program ini 🙏 ")
        print('''
```

```
        --------------------------------------------------------------------
--------
                Rasa ingin menyerah itu wajar, tetapi menyerah bukan keputusan
                 yang tepat untuk seorang manusia super tangguh sepertimu.
                       Tetaplah melangkah, walau harus merubah arah

                                    -Martabak Legit Group-
        --------------------------------------------------------------------
--------


        ''')
        d = input("tekan enter untuk menutup program...")

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

2.5 File divideAndConquer.py

```
import function as f

def Sstrip(distance, array, n, d, titikSebelumnya, count):
    if (len(array) % 2 == 0):
        mid = len(array) // 2
        t = (array[mid][0] + array[mid+1][0])/2
    else:
        mid = len(array) // 2
        t = array[mid+1][0]
    A = []
    for i in range(n):
        if ((array[i][0] >= (t-distance)) and (array[i][0] <= (distance+t))):
            A.append(array[i])

    for i in range(len(A)):
        for j in range(i+1,len(A)):
            if(abs(A[i][0]-A[j][0])<distance):
                B = [A[i], A[j]]
                jarakBaru, titikBaru, count = f.nilaiEuclidean(B,d,count)
                if (jarakBaru < distance):
                    distance = jarakBaru
                    titikSebelumnya = B
    return distance, titikSebelumnya, count

def FindClosestPair(array, n, d, count):
    distance = 0
    titik = []
    if n==2:
        distance, titik, count = f.nilaiEuclidean(array, d, count)
```

```
    elif n==3:
        arr1 = [array[0], array[1]]
        arr2 = [array[0], array[2]]
        arr3 = [array[1], array[2]]
        d1, titik1, count = f.nilaiEuclidean(arr1, d, count)
        d2, titik2, count = f.nilaiEuclidean(arr2, d, count)
        d3, titik3, count = f.nilaiEuclidean(arr3, d, count)

        if (d1 <= d2 and d1 <= d3):
            distance = d1
            titik = titik1
        elif (d2 <= d1 and d2 <= d3):
            distance = d2
            titik = titik2
        elif (d3 <= d1 and d3 <= d2):
            distance = d3
            titik = titik3

    else:
        kiri, kanan = f.divide(array, n, d)
        dKiri, titikKiri, count = FindClosestPair(kiri, len(kiri), d, count)
        dKanan, titikKanan, count = FindClosestPair(kanan, len(kanan), d,
count)
        distance = f.minimum(dKiri, dKanan)
        titik = []
        if (distance == dKiri):
            titik = titikKiri
        elif (distance == dKanan):
            titik = titikKanan
        distance, titik, count = Sstrip(distance, array, n, d, titik, count)
    return distance, titik, count
```

2.6 File bruteForce.py

```
import function as f

def bruteForce(array, n, d, count):
    a = 9999999999999999

    titik = []
    for i in range (n):
        for j in range (n):
            if (i!=j):
                matriks = f.buatPembanding(array, i, j, d)
                eucledian, matriks, count = f.nilaiEuclidean(matriks, d,
count)
                if (eucledian < a):
                    a = eucledian
```

```
                    titik = [array[i], array[j]]
    return a, titik, count
```

BAB III

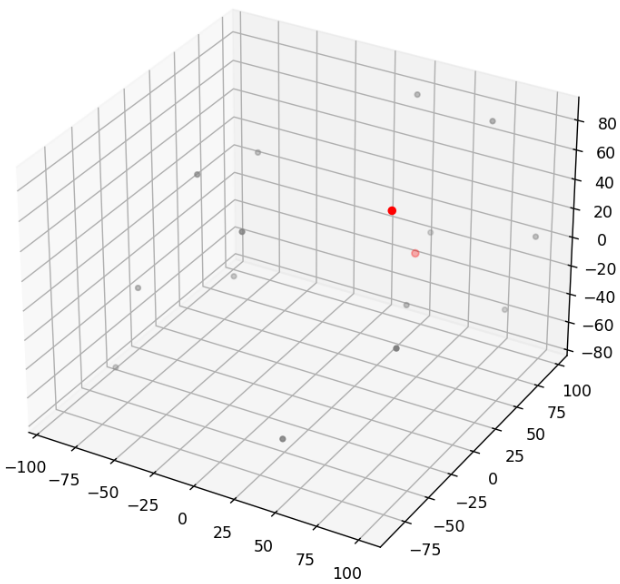
Uji Coba Program

3.1 Hasil Pengujian dengan n = 16 dan d = 3

Keluaran dalam terminal:

```
------------------------------
Nama : Agsha Athalla Nurkareem
NIM : 13521027
Mata Kuliah : IF2211
------------------------------


Masukkan banyaknya titik: 16
Masukkan banyaknya dimensi: 3


Processor : Intel64 Family 6 Model 142 Stepping 12, GenuineIntel

-------------------------------------------------------------------------

                        ALGORITMA DIVIDE AND CONQUER



jarak terdekat =  39.538182924863904
titik =  [[58.54, 4.109, 60.01], [64.163, 20.163, 24.318]]
execute time =  0.01529693603515625  seconds
euclidean count =  92


-------------------------------------------------------------------------

                            ALGORITMA BRUTE FORCE

jarak brute force =  39.538182924863904
titik brute force =  [[58.54, 4.109, 60.01], [64.163, 20.163, 24.318]]
execute time brute force =  0.0018870830535888672  seconds
euclidean count =  240
```

Keluaran dalam visualisasi:

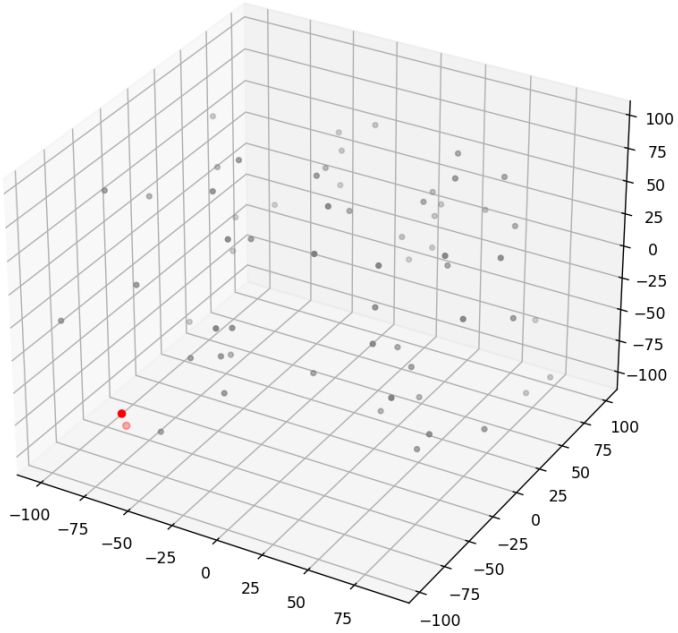


3.2 Hasil Pengujian dengan n = 64 dan d = 3

Keluaran dalam terminal:

```
 ______ _           _    _____ _                  _     _____      _
|  ____(_)         | |  / ____| |                | |   |  __ \    (_)
| |__   _ _ __   __| | | |    | | ___  ___  ___| |_  | |__) |_ _ _ _ __
|  __| | | '_ \ / _` | | |    | |/ _ \/ __|/ _ \ __| |  ___/ _` | | '__|
| |    | | | | | (_| | | |____| | (_) \__ \  __/\__ \ |_  | |  | (_| | | |
|_|    |_|_| |_|\__,_|  \_____|_|\___/|___/\___||___/\__| |_|   \__,_|_|_|

                        ------------------------------
                        Nama : Agsha Athalla Nurkareem
                        NIM : 13521027
                        Mata Kuliah : IF2211
                        ------------------------------


Masukkan banyaknya titik: 64
Masukkan banyaknya dimensi: 3


Processor : Intel64 Family 6 Model 142 Stepping 12, GenuineIntel

------------------------------------------------------------------------

                      ALGORITMA DIVIDE AND CONQUER



jarak terdekat =  8.26242095272324
titik =  [[-62.362, -93.919, -55.652], [-59.602, -94.415, -63.424]]
execute time =  0.0018360614776611328  seconds
euclidean count =  539


------------------------------------------------------------------------

                           ALGORITMA BRUTE FORCE

jarak brute force =  8.26242095272324
titik brute force =  [[-62.362, -93.919, -55.652], [-59.602, -94.415, -63.424]]
execute time brute force =  0.019185543060302734  seconds
euclidean count =  4032
```

Keluaran dalam visualisasi:

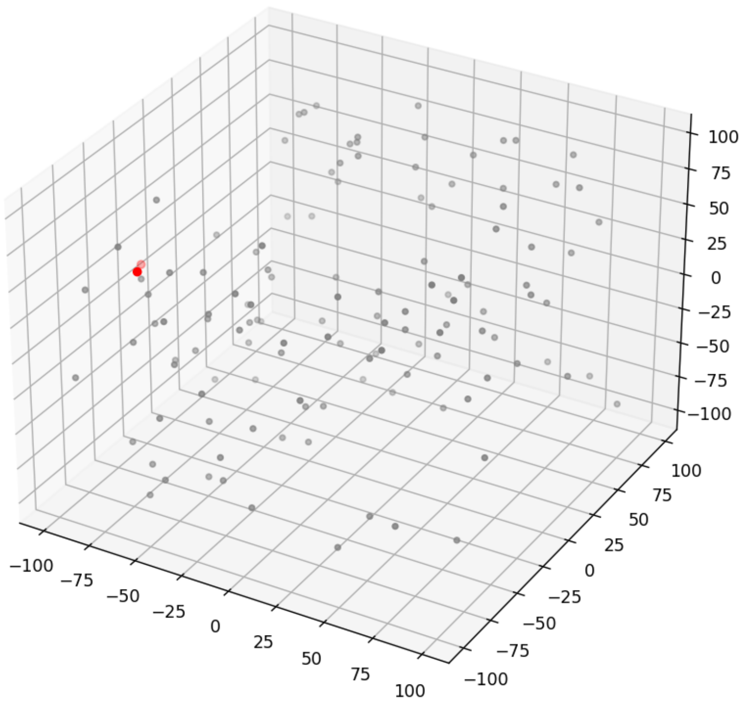


3.3 Hasil Pengujian dengan n = 128 dan d = 3

Keluaran dalam terminal:

```
------------------------------
Nama : Agsha Athalla Nurkareem
NIM : 13521027
Mata Kuliah : IF2211
------------------------------


Masukkan banyaknya titik: 128
Masukkan banyaknya dimensi: 3


Processor : Intel64 Family 6 Model 142 Stepping 12, GenuineIntel

------------------------------------------------------------------------
                        ALGORITMA DIVIDE AND CONQUER



jarak terdekat =  5.808620748508201
titik =  [[-98.853, -27.365, 18.592], [-97.438, -32.888, 17.481]]
execute time =  0.0  seconds
euclidean count =  1683



------------------------------------------------------------------------
                            ALGORITMA BRUTE FORCE

jarak brute force =  5.808620748508201
titik brute force =  [[-98.853, -27.365, 18.592], [-97.438, -32.888, 17.481]]
execute time brute force =  0.11769723892211914  seconds
euclidean count =  16256
```

Keluaran dalam visualisasi:

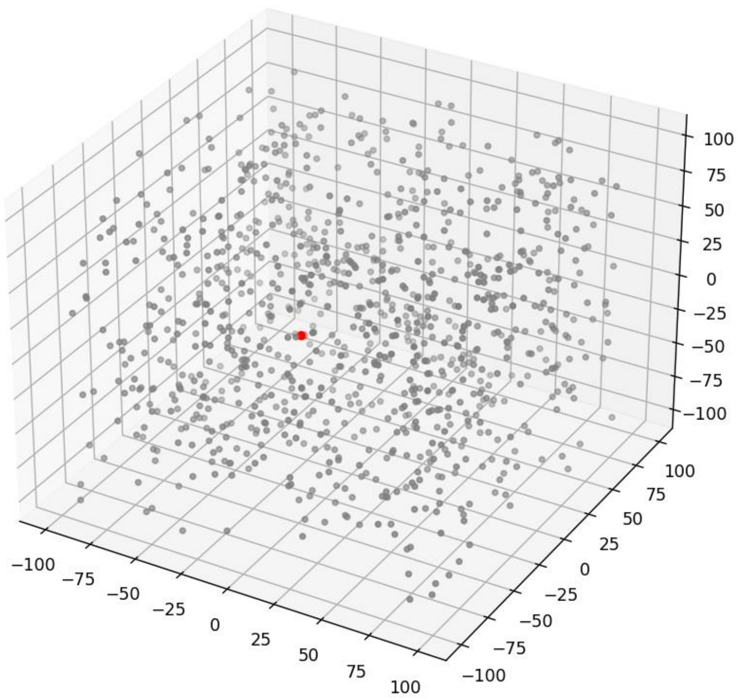


3.4 Hasil Pengujian dengan n = 1000 dan d = 3

Keluaran dalam terminal:

```
 ______ _        _   _____ _                _    _____     _
|  ____(_)      | | / ____| |              | |  |  __ \   (_)
| |__   _ _ __   __| || |    | | ___  ___  ___  ___| |_ | |__) |_ _ _ _ __
|  __| | | '_ \ / _` || |    | |/ _ \/ __|/ _ \/ __| __|| ___/ _` | | '__|
| |    | | | | | (_| || |____| | (_) \__ \  __/\__ \ |_ | |  | (_| | | |
|_|    |_|_| |_|\__,_| \_____|_|\___/|___/\___||___/\__||_|   \__,_|_|_|

                              ------------------------------
                              Nama : Agsha Athalla Nurkareem
                              NIM : 13521027
                              Mata Kuliah : IF2211
                              ------------------------------


Masukkan banyaknya titik: 1000
Masukkan banyaknya dimensi: 3


Processor : Intel64 Family 6 Model 142 Stepping 12, GenuineIntel

-------------------------------------------------------------------------

                        ALGORITMA DIVIDE AND CONQUER



jarak terdekat =  1.3354538554364244
titik =  [[-64.23, 60.999, -77.875], [-63.346, 61.964, -78.141]]
execute time =  0.07852649688720703  seconds
euclidean count =  40127


-------------------------------------------------------------------------

                            ALGORITMA BRUTE FORCE

jarak brute force =  1.3354538554364244
titik brute force =  [[-64.23, 60.999, -77.875], [-63.346, 61.964, -78.141]]
execute time brute force =  3.447863817214966  seconds
euclidean count =  999000
```

Keluaran dalam visualisasi:



3.5 Hasil Pengujian dengan n = 50 dan d = 6

Keluaran dalam terminal:

3.6 Hasil Pengujian dengan n = 100 dan d = 10

Keluaran dalam terminal:

BAB 4

LAMPIRAN

4.1 Tautan *repository* GitHub

https://github.com/agshaathalla/Tucil2_13521027.git

4.2 Cek list

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