

TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

Mencari Pasangan Titik Terdekat 3D

dengan

Algoritma Divide and Conquer



Disusun oleh:

13521155 - Kandida Edgina Gunawan

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

DAFTAR ISI	2
BAB 1 : Deskripsi Masalah dan Algoritma	3
Algoritma Divide and Conquer	
Masalah Pencarian Pasangan Titik Terdekat	3
Algoritma Penyelesaian Masalah dengan Pendekatan Divide and Conquer	4
BAB 2 : Source Code Program	5
Repository Program	5
Source Code Program	5
BAB 3 : Implementasi Algoritma dalam Bahasa Python	11
main.py	11
brute_force.py	12
divide_and_conquer.py	
visualizer.py	12
BAB 4 : Masukan dan Luaran Program	13
Masukan dan Luaran Program	13

BAB 1

DESKRIPSI MASALAH DAN ALGORITMA

1.1. Algoritma Divide and Conquer

Algoritma *Divide and Conquer* merupakan gabungan dari tiga proses utama, yaitu *divide*, *conquer*, dan *combine* yang digunakan untuk menyelesaikan suatu persoalan yang besar. *Divide* artinya membagi suatu persoalan yang skalanya cukup besar menjadi beberapa upa-persoalan yang sifatnya mirip dengan persoalan semula, namun dengan skala yang lebih kecil sehingga metode *Divide and Conquer* lebih natural diungkapkan dalam skema rekursif. *Conquer* berarti menyelesaikan tiap upa-persoalan baik secara langsung (persoalan sudah berukuran kecil) maupun tidak langsung dengan cara rekursif (persoalan masih berukuran besar). *Combine* memiliki arti menggabungkan solusi tiap-tiap upa-persoalan sehingga membentuk solusi persoalan semula.

Terdapat banyak persoalan yang dapat diselesaikan dengan menggunakan pendekatan algoritma *Divide and Conquer*. Persoalan-persoalan yang dapat diselesaikan di antaranya, yaitu persoalan mencari nilai minimum dan maksimum di dalam suatu larik, menghitung perpangkatan, persoalan pengurutan (*merge sort* dan *quicksort*), pencarian sepasang titik terdekat (*closest pair problem*), *Convex Hull*, perkalian matriks, perkalian bilangan bulat besar, dan perkalian dua buah polinom. Pada tugas kecil 2 ini, persoalan yang ingin diselesaikan adalah persoalan pencarian pasangan titik terdekat.

1.2. Masalah Pencarian Pasangan Titik Terdekat

Pada persoalan pencarian pasangan titik terdekat, diberikan himpunan titik, P , yang terdiri dari n buah titik pada bidang 3 dimensi. Dari n buah titik tersebut akan dipilih sepasang titik di dalam P yang jaraknya terdekat satu sama lain. Jarak dua buah titik $p_1 = (x_1, y_1, z_1)$ dan $p_2 = (x_2, y_2, z_2)$ dihitung dengan rumus *Euclidean*

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

Persoalan pencarian pasangan titik terdekat dapat diselesaikan dengan pendekatan algoritma *brute force*. Dengan pendekatan *brute force*, kita akan menghitung jarak setiap pasang titik. Terdapat sebanyak $C(n, 2) = n(n-1)/2$ pasang titik yang harus dihitung jaraknya. Kemudian, kita akan membandingkan jarak dari tiap pasang titik tersebut kemudian memilih pasangan titik yang mempunyai jarak terkecil

sebagai solusinya. Kompleksitas algoritma dari pendekatan *brute force* ini adalah $O(n^2)$.

Selain menggunakan pendekatan *brute force*, persoalan pencarian pasangan titik terdekat dapat pula diselesaikan dengan pendekatan *divide and conquer* yang akan dijelaskan di bawah ini.

1.3. Algoritma Penyelesaian Masalah dengan Pendekatan Divide and Conquer

Praproses : titik-titik di dalam P diurut menaik berdasarkan nilai koordinatnya menggunakan skema *sorting merge sort*

1. *Solve*: Jika $n \leq 3$, gunakan *brute force* untuk mencari sepasang titik dengan jarak terdekat
2. *Divide* : Bagi himpunan titik ke dalam dua bagian, S1, dan S2, setiap bagian mempunyai jumlah titik yang sama. L adalah garis maya yang membagi dua himpunan titik ke dalam dua sub-himpunan, masing-masing $n/2$ titik.
Garis maya L dapat dihampiri sebagai $y = xn/2$
3. *Conquer* : Secara rekursif, terapkan algoritma *Divide and Conquer* pada masing-masing bagian untuk mencari sepasang titik terdekat.
4. *Combine* : Pasangan titik yang jaraknya terdekat ada tiga kemungkinan letaknya:
 - a. Pasangan titik terdekat terdapat di dalam bagian S1.
 - b. Pasangan titik terdekat terdapat di dalam bagian S2
 - c. Pasangan titik terdekat dipisahkan oleh garis batas L, yaitu satu titik di S1 dan satu titik di S2. Bagian yang pasangan titiknya dipisahkan oleh garis batas L ini kemudian akan kita sebut sebagai bagian S3.

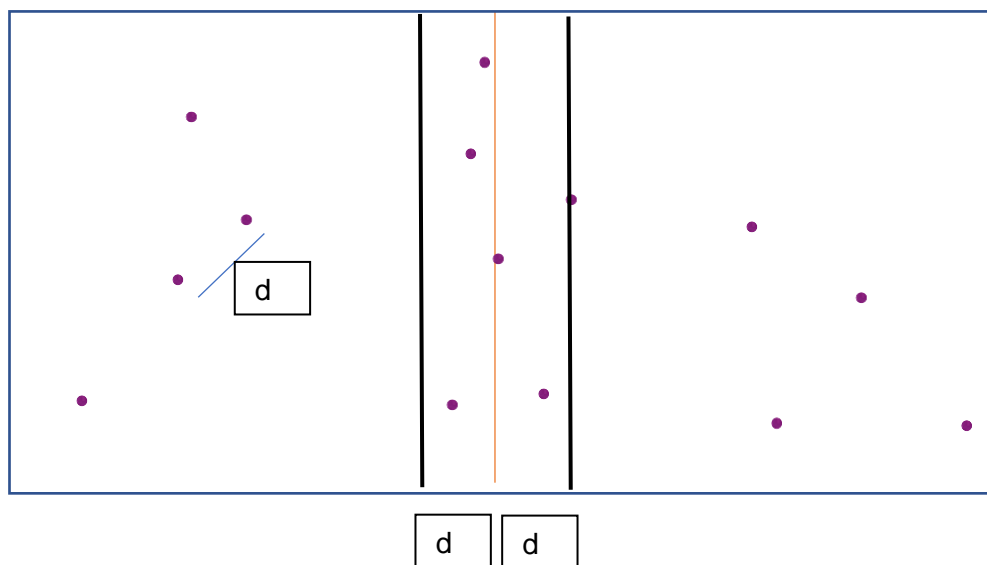
Saat *combine*, akan dibandingkan pasangan titik terdekat yang terdapat di dalam bagian S1 dan bagian S2. Setelah didapatkan jarak terdekat dari 2 titik pada bagian S1 dan S2, kita akan membandingkan pasangan titik tersebut dengan pasangan titik yang salah satu titiknya terdapat di S1 dan satu titik lainnya yang terdapat di S2 (pasangan titik yang terdapat pada bagian S3) untuk mencari pasangan titik terdekat yang terdapat pada himpunan P.

Untuk mencari titik-titik yang berada pada bagian S3, kita akan mengambil titik-titik dari himpunan P yang jarak posisinya di sumbu x terhadap titik tengah antara S1 dan S2 kurang dari jarak terdekat sementara yang didapatkan dengan membandingkan jarak terdekat pasangan pada bagian S1 dan bagian S2.

Ilustrasi dari titik-titik yang terdapat pada bagian S3 dapat dilihat pada gambar di bawah ini.

Karena jarak antartitik yang terdekat untuk bagian S1 dan S2 adalah d , maka batasan daerah S3 adalah daerah yang dibatasi oleh d satuan ke kanan dan d satuan ke kiri dari garis khayal yang telah ditetapkan.

Pada bagian S3, kemudian dicari juga pasangan titik dengan jarak terdekat dengan menggunakan *brute force* (karena jumlah titiknya sedikit) yang kemudian akan dibandingkan dengan d .



BAB 2

3.1. Link Github Repository

Link Repository dari Tugas Kecil Strategi Algoritma yang kedua ini dapat dilihat pada

https://github.com/kandidagunawan/Tucil2_13521155

3.2. Source Code Program

3.2.1.main.py

```

1 import random
2 from basic_functions import *
3 from divide_and_conquer import *
4 from timeout import default_timer as timer
5 from visualizer import plotPoints
6
7 print('
8
9 print(
10
11
12 print(
13
14 print('
15 print(
16
17 print('
18 print('
19 print(
20
21 print('
22 print(
23
24 print('
25 print('
26 print('\n')
27 print('.....')
28 print('\n')
29 running = True
30 while (running == True):
31     validate = False
32     while (validate == False):
33         dimensi = input("Masukkan dimensi dari titik yang diinginkan: ")
34         try:
35             dimensi = int(dimensi)
36             if (dimensi <= 0):
37                 raise ValueError
38         except:
39             print("Masukan hanya boleh berupa bilangan bulat positif!")
40         else:
41             validate = True
42     validate = False
43
44     validate = False
45     while (validate == False):
46         n = input("Masukkan jumlah titik yang diinginkan (n): ")
47         try:
48             n = int(n)
49             if (n < 2):
50                 raise ValueError
51         except:
52             print("Masukan hanya boleh berupa bilangan bulat dengan nilai minimum 2!")
53         else:
54             validate = True
55     list = []
56     for i in range(n):
57         tuple = ()
58         for j in range(dimensi):
59             temp = random.uniform(-1000, 1000)
60             tuple += (temp,)
61         list.append(tuple)
62
63     # PRINT EVERY POINT
64     # for i in range(n):
65     #     print('Point', i+1, ":", list[i])
66
67     start1 = timer()
68     distance, listOffPoints1, numEucl1 = bruteForce(list)
69     end1 = timer()
70
71     start2 = timer()
72     mergesort(list, 0, n-1, 0)
73     d1, listOffPoints2, numEucl2 = closestPair(list, n)
74     end2 = timer()
75
76     print('\n')
77     print("BRUTE FORCE")
78     print("Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah ", distance)
79     print("Jarak tersebut merupakan jarak antara 2 titik yaitu:")
80     print(PasanganTitik(listOffPoints1))
81     print("Waktu yang dibutuhkan oleh brute force: ")
82     print(end1-start1, ' detik')
83     print("Banyak perhitungan Euclidean Distance:", numEucl1)
84
85     print('\n')
86     print("DIVIDE AND CONQUER")
87     print("Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah ", d1)
88     print("Jarak tersebut merupakan jarak antara 2 titik yaitu:")
89     print(PasanganTitik(listOffPoints2))
90
91     print("Waktu yang dibutuhkan oleh divide and conquer:")

```

```

90     print(end2-start2, ' detik')
91     print('Banyak perhitungan Euclidean Distance:', numEuc2)
92     print('\n')
93     if (dimensi == 3):
94         plotPoints(list, listOfPoints2)
95     validate = False
96     while (validate == False):
97         stop = input("Apakah kamu ingin keluar dari program? (y/n)")
98         if (stop == "y" or stop == "n"):
99             validate = True
100     print('\n')
101     print('\n')
102     if (stop == "y"):
103         running = False
104

```

3.2.2 basic_functions.py

```

1  def euclideanDistance(tuple1, tuple2):
2      distance2 = 0
3      for i in range(len(tuple1)):
4          distance2 += ((tuple1[i] - tuple2[i])**2)
5      return (distance2 ** (1/2))
6
7
8  def bruteForce(list):
9      distance = euclideanDistance(list[0], list[1])
10     listOfPoints = []
11     point1 = list[0]
12     point2 = list[1]
13     points = [point1, point2]
14     numEuc = 0
15     for i in range(0, len(list)):
16         for j in range(i+1, len(list)):
17             temp = euclideanDistance(list[i], list[j])
18             numEuc += 1
19             if (temp < distance):
20                 distance = temp
21                 point1 = list[i]
22                 point2 = list[j]
23                 points = [point1, point2]
24                 listOfPoints = [points]
25             elif (temp == distance):
26                 listOfPoints.append(points)
27
28     return distance, listOfPoints, numEuc
29
30
31 def printPasanganTitik(list):
32     print("Berikut ini adalah pasangan titik dengan jarak terdekat:")
33     for i in range(len(list)):
34         print("Pasangan titik ke-", i+1)
35         for j in range(2):
36             print(list[i][j])
37

```

3.2.3 divide_and_conquer.py

```
from basic_functions import *

def merge(arr, left, middle, right, koordinat):
    n1 = middle - left + 1
    n2 = right - middle

    # Membuat 2 array temp
    L = [0 for i in range(n1)]
    R = [0 for i in range(n2)]

    for i in range(0, n1):
        L[i] = arr[left + i]

    for j in range(0, n2):
        R[j] = arr[middle + 1 + j]

    # Merge arr L dan R ke dalam arr
    i = 0 # Indeks dari array L
    j = 0 # Indeks dari array R
    k = left # Indeks dari array arr

    while i < n1 and j < n2:
        if L[i][koordinat] < R[j][koordinat]: # Membandingkan elemen ke-koordinat tuple
            arr[k] = L[i]
            i += 1
        # Jika elemen pertama tuple sama, cek elemen berikutnya sampai ada elemen yang berbeda
        elif L[i][koordinat] == R[j][koordinat]:
            temp = 0
            sama = True
            while (sama and temp < len(L[i])):
                if (L[i][temp] > R[j][temp]):
                    arr[k] = R[j]
                    j += 1
                    sama = False
                elif (L[i][temp] < R[j][temp]):
                    arr[k] = L[i]
                    i += 1
                    sama = False
                else:
                    temp += 1
            if (sama):
                arr[k] = L[i]
                i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1
```



```

# Copy sisa element dari L ke arr
while i < n1:
    arr[k] = L[i]
    i += 1
    k += 1

# Copy sisa element daari R ke arr
while j < n2:
    arr[k] = R[j]
    j += 1
    k += 1

def mergeSort(arr, left, right, koordinat):
    if (left < right):
        middle = left + (right-left) // 2
        mergeSort(arr, left, middle, koordinat)
        mergeSort(arr, middle + 1, right, koordinat)
        merge(arr, left, middle, right, koordinat)

def stripPoints(arr, distance, divider, middlePoint):
    points = []
    for i in range(middlePoint, len(arr)):
        if (arr[i][0] - divider[0] > distance):
            break
        else:
            points.append(arr[i])
    for i in range(middlePoint-1, -1, -1):
        if (divider[0]-arr[i][0] > distance):
            break
        else:
            points.append(arr[i])
    return points

```

```

def closestPair(arr, n):
    if (n <= 3):
        return bruteForce(arr) # base case
    else:
        middle = n//2
        distance1, listOfPoints1, numEuc1 = closestPair( # mencari closest pair pada s1
            arr[:middle], middle)
        distance2, listOfPoints2, numEuc2 = closestPair( # mencari closest pair pada s2
            arr[middle:n], n-middle)
        if (distance1 < distance2):
            distance = distance1
            listOfPoints = listOfPoints1
        elif (distance1 == distance2):
            listOfPoints = listOfPoints1 + listOfPoints2
        else:
            distance = distance2
            listOfPoints = listOfPoints2

        strips = stripPoints(arr, distance, arr[middle], middle)

        if (len(strips) >= 2):
            distanceStrip, listOfPointStrip, numEuc = bruteForce(
                strips) # mencari closest pair pada s3
            if (distanceStrip < distance):
                distance = distanceStrip
                listOfPoints = listOfPointStrip
            elif (distanceStrip == distance):
                for x in listOfPointStrip:
                    if (x in listOfPoints):
                        continue
                    else:
                        listOfPoints.append(x)
            return distance, listOfPoints, (numEuc + numEuc1 + numEuc2)
        else:
            return distance, listOfPoints, (numEuc1 + numEuc2)

```

3.2.4. visualizer.py

```

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import itertools

def plotPoints(list, listOfPoints):
    fig = plt.figure(figsize=(4, 4))
    ax = fig.add_subplot(111, projection='3d')
    ax.set_title("Points in 3D")
    ax.set_xlabel("x-axis")
    ax.set_ylabel("y-axis")
    ax.set_zlabel("z-axis")
    colors = itertools.cycle(['orange', 'green', 'red', 'purple',
                              'brown', 'pink', 'gray', 'olive', 'cyan'])
    for i in range(len(listOfPoints)):
        c1 = next(colors)
        for j in range(2):
            x = listOfPoints[i][j][0]
            y = listOfPoints[i][j][1]
            z = listOfPoints[i][j][2]
            ax.scatter(x, y, z, c=c1)
    for i in range(len(list)):
        x = list[i][0]
        y = list[i][1]
        z = list[i][2]
        for j in range(len(listOfPoints)):
            if (list[i] not in listOfPoints[j]):
                ax.scatter(x, y, z, c='blue')
    plt.show()

```

BAB 3

IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

Dalam pembuatan program ini, penulis menggunakan bahasan pemrograman *Python*. Struktur dari program ini terbagi menjadi 4 *file*, yaitu *main.py*, *divide_and_conquer.py*, *visualizer.py*, dan *basic_functions.py*.

2.1. main.py

File ini berisi kode yang digunakan untuk menggabungkan fungsi-fungsi yang berada pada *file* lain dan menjalankan program utama. Pada *file* ini diatur pula mengenai *input handling* dan tampilan dari program utama.

2.2. basic_functions.py

Fungsi	Deskripsi
<code>euclideanDistance(tuple1, tuple2)</code>	Fungsi ini berguna untuk memberikan jarak antara 2 titik, yaitu <code>tuple1</code> dan <code>tuple2</code>
<code>bruteForce(list)</code>	Fungsi ini berguna untuk mencari pasangan titik dengan jarak <i>euclidean distance</i> terkecil dengan menggunakan pendekatan <i>brute force</i> . Fungsi ini akan <i>me-return</i> tiga keluaran yaitu jarak terdekat sepasang titik yang berada pada list, list yang berisi pasangan titik dengan jarak terdekat, serta banyaknya perhitungan <i>euclidean distance</i> yang dilakukan.
<code>printPasanganTitik(list)</code>	Fungsi ini berguna untuk mencetak pasangan-pasangan titik dengan jarak terdekat ke layar

2.3. divide_and_conquer.py

Fungsi	Deskripsi
<code>merge(arr, left, middle, right)</code>	Fungsi ini akan menggabungkan 2 list yang sudah terurut menjadi 1 list yang terurut membesar
<code>mergeSort(arr, left, right)</code>	Fungsi ini akan melakukan <i>divide</i> yaitu membagi list menjadi 2 bagian dan melakukan proses <code>mergeSort</code> untuk kedua bagian tersebut. Kemudian, hasil <code>mergeSort</code> dari kedua list tersebut akan digabungkan untuk mendapatkan hasil suatu list yang sudah terurut.
<code>stripPoints(arr, distance, divider, middlePoint)</code>	Fungsi ini akan menghasilkan himpunan titik-titik yang berada pada bagian S3
<code>closestPair(arr, n)</code>	Fungsi ini akan melakukan rekursif untuk menemukan pasangan titik dengan jarak terdekat pada bagian S1 dan S2 dengan <i>base case</i> nya adalah jumlah titik pada list yang

	<p>kurang dari atau sama dengan 3. Apabila jumlah dari titik pada suatu list kurang dari atau sama dengan 3, akan dilakukan <i>brute force</i> untuk menemukan pasangan titik dengan jarak terdekat.</p> <p>Setelah ditemukan pasangan titik dengan jarak terdekat pada bagian S1 dan S2, kita akan mencari pasangan titik dengan jarak terdekat pada S3 dan membandingkannya dengan pasangan titik yang sebelumnya sudah kita dapatkan pada S1 dan S2. Fungsi ini akan me-<i>return</i> jarak terdekat pasangan titik, <i>list</i> yang berisi pasangan titik dengan jarak terdekat, serta banyaknya operasi <i>brute force</i> yang dilakukan.</p>
--	--

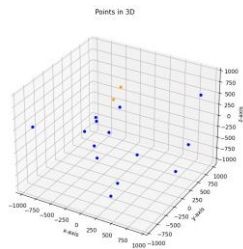
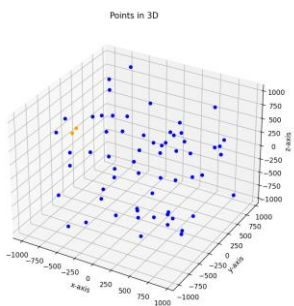
2.4. visualizer.py

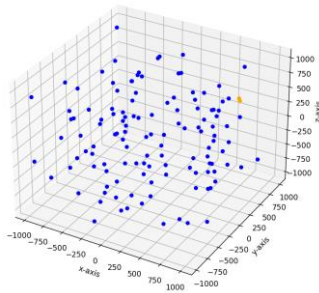
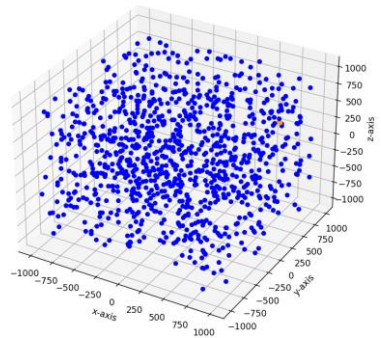
Fungsi	Deskripsi
plotPoints(list, listOfPoints)	Fungsi ini akan menampilkan ke layar visualisasi 3 dimensi dari titik-titik yang berada pada himpunan P. Pasangan titik yang memiliki jarak terdekat akan memiliki warna yang berbeda dengan titik-titik yang lainnya.

BAB 4

Masukan dan Luaran Program

Spek laptop : AMD Ryzen 7 3700U, RAM : 8GB

No	Test case	Masukan dan Keluaran
1	Dimensi : 3 Jumlah titik: 16	<pre> BRUTE FORCE Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 245.8938937925579 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (13.718265050902232, 47.98786702097944, 989.9636668897854) (-35.737822616059816, -72.55653882448462, 701.3831407329728) Waktu yang dibutuhkan oleh brute force: 0.0002989000640809536 detik Banyak perhitungan Euclidean Distance: 120 DIVIDE AND CONQUER Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 245.8938937925579 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (-35.737822616059816, -72.55653882448462, 701.3831407329728) (13.718265050902232, 47.98786702097944, 989.9636668897854) Waktu yang dibutuhkan oleh divide and conquer: 0.00034509995820926 detik Banyak perhitungan Euclidean Distance: 38 </pre> 
2	Dimensi : 3 Jumlah titik : 64	<pre> Masukkan dimensi dari titik yang diinginkan: 3 Masukkan jumlah titik yang diinginkan (n): 64 BRUTE FORCE Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 107.17120014133036 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (-664.180111576597, -393.7480193855369, 620.4753974337805) (-655.2364217466427, -497.919361591614, 596.9380229354294) Waktu yang dibutuhkan oleh brute force: 0.8004067800007760525 detik Banyak perhitungan Euclidean Distance: 2016 DIVIDE AND CONQUER Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 107.17120014133036 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (-655.2364217466427, -497.919361591614, 596.9380229354294) (-664.180111576597, -393.7480193855369, 620.4753974337805) Waktu yang dibutuhkan oleh divide and conquer: 0.8029769001994282807 detik Banyak perhitungan Euclidean Distance: 432 </pre> 

3	<p>Dimensi : 3</p> <p>Jumlah titik : 128</p>	<pre> Masukkan dimensi dari titik yang diinginkan: 3 Masukkan jumlah titik yang diinginkan (n): 128 BRUTE FORCE Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 74.81238924092517 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (618.4215871788083, 940.0740440555451, 198.6342802377742) (574.4229267458397, 999.8756257585001, 189.426550340301) Waktu yang dibutuhkan oleh brute force: 0.01509390974942207 detik Banyak perhitungan Euclidean Distance: 8128 DIVIDE AND CONQUER Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 74.81238924092517 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (618.4215871788083, 940.0740440555451, 198.6342802377742) (574.4229267458397, 999.8756257585001, 189.426550340301) Waktu yang dibutuhkan oleh divide and conquer: 0.005624499870464206 detik Banyak perhitungan Euclidean Distance: 953 </pre> <p>Points in 3D</p> 
4	<p>Dimensi 3</p> <p>Jumlah titik : 1000</p>	<pre> Masukkan dimensi dari titik yang diinginkan: 3 Masukkan jumlah titik yang diinginkan (n): 1000 BRUTE FORCE Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 16.3205419021086 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (774.0028516052716, 641.729857603656, 349.43245458613546) (765.2374774936372, 631.3547789172346, 358.48153635783) Waktu yang dibutuhkan oleh brute force: 1.10896490002051 detik Banyak perhitungan Euclidean Distance: 499500 DIVIDE AND CONQUER Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 16.3205419021086 Jarak tersebut merupakan jarak antara 2 titik yaitu: Berikut ini adalah pasangan titik dengan jarak terdekat: Pasangan titik ke- 1 (774.0028516052716, 641.729857603656, 349.43245458613546) (765.2374774936372, 631.3547789172346, 358.48153635783) Waktu yang dibutuhkan oleh divide and conquer: 0.07240879905220131 detik Banyak perhitungan Euclidean Distance: 19319 </pre> <p>Points in 3D</p> 

5	<p>BONUS</p> <p>Dimensi : 4</p> <p>Jumlah titik : 16</p>	<p>Masukkan dimensi dari titik yang diinginkan: 4</p> <p>Masukkan jumlah titik yang diinginkan (n): 16</p> <p>BRUTE FORCE</p> <p>Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 590.0711529962135</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-521.6514776164692, -101.19967622507329, 214.65327481524469, 118.69927395588479)</p> <p>(-235.28079491639153, 286.5115388536808, 63.33164114758665, -186.19017252345714)</p> <p>Waktu yang dibutuhkan oleh brute force:</p> <p>0.0003306001890450716 detik</p> <p>Banyak perhitungan Euclidean Distance: 120</p> <p>DIVIDE AND CONQUER</p> <p>Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 590.0711529962135</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-235.28079491639153, 286.5115388536808, 63.33164114758665, -186.19017252345714)</p> <p>(-521.6514776164692, -101.19967622507329, 214.65327481524469, 118.69927395588479)</p> <p>Waktu yang dibutuhkan oleh divide and conquer:</p> <p>0.0008984999731183052 detik</p> <p>Banyak perhitungan Euclidean Distance: 97</p>
6	<p>BONUS</p> <p>Dimensi : 5</p> <p>Jumlah titik : 64</p>	<p>Masukkan dimensi dari titik yang diinginkan: 5</p> <p>Masukkan jumlah titik yang diinginkan (n): 64</p> <p>BRUTE FORCE</p> <p>Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 282.0253179700394</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(186.900104090056313, -274.25587675978003, 691.0595438768048, -515.2515944082061, 863.5302045726903)</p> <p>(209.93675108283178, -90.62472129886316, 632.909226605726, -611.3238812062893, 602.7655816823374)</p> <p>Waktu yang dibutuhkan oleh brute force:</p> <p>0.006018700078129768 detik</p> <p>Banyak perhitungan Euclidean Distance: 2016</p> <p>DIVIDE AND CONQUER</p> <p>Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 282.0253179700394</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(186.900104090056313, -274.25587675978003, 691.0595438768048, -515.2515944082061, 863.5302045726903)</p> <p>(209.93675108283178, -90.62472129886316, 632.909226605726, -611.3238812062893, 602.7655816823374)</p> <p>Waktu yang dibutuhkan oleh divide and conquer:</p> <p>0.002493900014087558 detik</p> <p>Banyak perhitungan Euclidean Distance: 379</p>
7	<p>BONUS</p> <p>Dimensi : 6</p> <p>Jumlah titik 128</p>	<p>Masukkan dimensi dari titik yang diinginkan: 6</p> <p>Masukkan jumlah titik yang diinginkan (n): 128</p> <p>BRUTE FORCE</p> <p>Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 389.26938140807505</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-513.6231268205731, 102.40216232530565, 693.1590638561006, 581.6935719137116, 637.2353512075547, -691.3447684113055)</p> <p>(-388.96662811023646, -404.60014468345321, 678.0519140153656, 277.70976112437006, 728.7124881824336, -686.9748694346982)</p> <p>Waktu yang dibutuhkan oleh brute force:</p> <p>0.030768300173804104 detik</p> <p>Banyak perhitungan Euclidean Distance: 8128</p> <p>DIVIDE AND CONQUER</p> <p>Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 389.26938140807505</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-513.6231268205731, 102.40216232530565, 693.1590638561006, 581.6935719137116, 637.2353512075547, -691.3447684113055)</p> <p>(-388.96662811023646, -404.60014468345321, 678.0519140153656, 277.70976112437006, 728.7124881824336, -686.9748694346982)</p> <p>Waktu yang dibutuhkan oleh divide and conquer:</p> <p>0.008549299789592628 detik</p> <p>Banyak perhitungan Euclidean Distance: 1394</p>
8	<p>BONUS</p> <p>Dimensi 7</p> <p>Jumlah titik : 1000</p>	<p>Masukkan dimensi dari titik yang diinginkan: 7</p> <p>Masukkan jumlah titik yang diinginkan (n): 1000</p> <p>BRUTE FORCE</p> <p>Jarak terdekat antara 2 titik berdasarkan BRUTE FORCE adalah 255.0118564085227</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-041.02307030338255, -510.52109000783708, -201.17899284687928, -909.5698629981991, 403.4068226242015, -131.17073732862004, 579.5425180050124)</p> <p>(-124.237024133016, -608.430782005042, -191.66081267786763, -871.0519879005143, 244.5537060837921, -219.59524067789076, 708.697427691816)</p> <p>Waktu yang dibutuhkan oleh brute force:</p> <p>1.966530900077035 detik</p> <p>Banyak perhitungan Euclidean Distance: 499500</p> <p>DIVIDE AND CONQUER</p> <p>Jarak terdekat antara 2 titik berdasarkan DIVIDE AND CONQUER adalah 279.3866394270065</p> <p>Jarak tersebut merupakan jarak antara 2 titik yaitu:</p> <p>Berikut ini adalah pasangan titik dengan jarak terdekat:</p> <p>Pasangan titik ke- 1</p> <p>(-545.1360067072811, -560.3609286677108, 112.16635418370765, -409.4007171353101, -909.6022379398541, -640.5110501405865, -67.27281401197369)</p> <p>(-545.9187629850506, -638.097521116781, 268.9207118389113, -277.42764007974927, -952.99623911089, -514.3097176704714, 5.33655154211283)</p> <p>Waktu yang dibutuhkan oleh divide and conquer:</p> <p>0.11222000000099378 detik</p> <p>Banyak perhitungan Euclidean Distance: 19018</p>

LAMPIRAN

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