

LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

MENCARI PASANGAN TITIK TERDEKAT DENGAN
ALGORITMA DIVIDE AND CONQUER

Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T, M.Sc.



Oleh :
Satria Octavianus Nababan / 13521168

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023

DAFTAR ISI

DAFTAR ISI.....	2
BAB I	3
DESKRIPSI MASALAH.....	3
BAB II.....	4
ALGORITMA.....	4
2.1 Algoritma Divide and Conquer	4
2.2 Penerapan Algoritma <i>Divide and Conquer</i> Pada Pencarian Pasangan Titik Terdekat.....	5
BAB III.....	6
SOURCE CODE PROGRAM.....	6
3.1 main.py.....	6
3.2 divideandconquer.py	8
3.3 bruteforce.py	11
3.4 visualisasi.py	12
BAB IV	14
EKSPERIMEN.....	14
4.1 Points = 16	14
4.2 Points = 64	16
4.3 Points = 128	18
4.4 Points = 1000	20
4.5 Kasus Points atau Dimensi Tidak Valid	22
LAMPIRAN.....	23
Link repository github :	23
Tabel check list :	23
DAFTAR PUSTAKA	24

BAB I

DESKRIPSI MASALAH

Mencari sepasang 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 diminta mengembangkan algoritma mencari sepasang 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 $P_1 = (x_1, y_1, z_1)$ dan $P_2 = (x_2, y_2, z_2)$ dihitung dengan rumus *Euclidean* berikut:

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

Kemudian mencari sepasang titik yang jaraknya terdekat satu sama lain dengan menerapkan algoritma *divide and conquer* untuk penyelesaiannya, dan perbandingannya dengan Algoritma *Brute Force*.

Masukan program:

- n
- titik-titik (dibangkitkan secara acak) dalam koordinat (x, y, z)

Luaran program

- sepasang titik yang jaraknya terdekat dan nilai jaraknya
- banyaknya operasi perhitungan rumus Euclidian
- waktu riil dalam detik (spesifikasikan komputer yang digunakan)
- Bonus 1 (Nilai = 7,5) penggambaran semua titik dalam bidang 3D, sepasang titik yang jaraknya terdekat ditunjukkan dengan warna yang berbeda dari titik lainnya.
- Bonus 2 (nilai = 7,5): Generalisasi program anda sehingga dapat mencari sepasang titik

terdekat untuk sekumpulan vektor di R_n , setiap vektor dinyatakan dalam bentuk $x = (x_1, x_2, \dots, x_n)$

BAB II

ALGORITMA

2.1 Algoritma Divide and Conquer

Algoritma *Divide and Conquer* adalah algoritma yang memecah persoalan menjadi beberapa upa-persoalan yang memiliki karakteristik yang sama dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Pembagian persoalan menjadi beberapa bagian tersebut diharapkan akan membantu mereduksi kompleksitas dari masalah sehingga lebih mudah untuk diselesaikan.

Setelah persoalan dibagi-bagi, selanjutnya melakukan *Conquer (solve)* yaitu menyelesaikan tiap upa-persoalan secara langsung jika sudah cukup kecil, dan menggabungkan hasil dari tiap upa-persoalan sehingga didapat solusi dari persoalan semula.

Skema Umum Algoritma *Divide and Conquer*

```
procedure DIVIDEandCONQUER(input P : problem, n : integer)
{ Menyelesaikan persoalan P dengan algoritma divide and conquer
  Masukan: masukan persoalan P berukuran n
  Luaran: solusi dari persoalan semula }
Deklarasi
  r : integer

Algoritma
  if  $n \leq n_0$  then {ukuran persoalan P sudah cukup kecil}
    SOLVE persoalan P yang berukuran n ini
  else
    DIVIDE menjadi r upa-persoalan,  $P_1, P_2, \dots, P_r$ , yang masing-masing berukuran  $n_1, n_2, \dots, n_r$ 
    for masing-masing  $P_1, P_2, \dots, P_r$ , do
      DIVIDEandCONQUER( $P_i, n_i$ )
    endfor
    COMBINE solusi dari  $P_1, P_2, \dots, P_r$  menjadi solusi persoalan semula
  endif
```

Kompleksitas algoritma *divide and conquer*:
$$T(n) = \begin{cases} g(n) & , n \leq n_0 \\ T(n_1) + T(n_2) \dots + T(n_r) + f(n) & , n > n_0 \end{cases}$$

Gambar 1.1 Skema Umum Algoritma *Divide and Conquer*

Selain itu, pada pemecahan masalah dalam tugas ini, saya juga menggunakan algoritma Quick Sort untuk mengurutkan titik-titik yang akan dicari jarak terdekatnya. Pemilihan skema pengurutan ini karena dianggap sebagai skema tercepat dan termasuk ke dalam pendekatan sulit membagi, mudah menggabung (*hard split/easy join*).

2.2 Penerapan Algoritma *Divide and Conquer* Pada Pencarian Pasangan Titik Terdekat

Berikut adalah langkah-langkah penerapan algoritma divide and conquer dalam pencarian titik terdekat :

1. Awalnya, program akan menerima masukan banyaknya titik dan dimensi yang akan dicari. Jika masukan valid, yaitu titik > 0 dan dimensi > 0 , maka masukan titik tersebut akan di random menggunakan prosedur *random_points*.
2. Selanjutnya, masukan titik yang telah *dirandom* akan diurutkan menggunakan pendekatan Quick Sort, yaitu mengurutkan titik berdasarkan referensi posisinya pada sumbu X.
3. Jika masukan titik hanya berjumlah 2 maka dapat langsung menghitung jaraknya dengan perhitungan *euclidean*.
4. Jika jumlah masukannya 3, akan dicari jarak dari setiap pasangan titik dan kemudian dibandingkan dan ditentukan pasangan titik dengan jarak terdekat.
5. Jika masukan titik lebih banyak (> 3), secara rekursif akan memilih elemen rata-rata pada titik X sebagai pivot dan membagi larik titik menjadi 2 bagian sama rata berdasarkan referensi pivot yang telah ditentukan. Lalu larik dibagi kembali dengan metode yang sama secara terus-menerus hingga hanya tersisa 2 atau 3 elemen.
6. Masing-masing dari daerah bagian larik dicari jarak terkecilnya, kemudian dibandingkan dan diambil nilai minimum dari hasil perbandingan.
7. Perbandingan ini terus dilakukan untuk setiap pembagian larik, hingga akhirnya didapat pasangan titik yang memiliki jarak terkecil sebagai hasil akhirnya.

BAB III

SOURCE CODE PROGRAM

Seluruh program diimplementasikan menggunakan bahasa python versi 3.11.0 pada sistem operasi Windows 10. Program terdiri atas file divideandconquer.py, bruteforce.py, visualisasi.py dan main.py.

3.1 main.py

File ini merupakan main program yang akan menerima *input*, menjalankan proses dan mengeluarkan *output* yang diharapkan.

```
from divideandconquer import *
from bruteforce import *
from visualisasi import *
import time
import platform
import numpy as np

points = int(input("Masukkan banyak titik : "))
dimensi = int(input("Masukkan dimensi : "))

if (points > 1 and dimensi > 0):
    arr_points = random_points(points,dimensi)
    quickSort(arr_points,0,points-1)
    if (dimensi == 3):
        count = 0
        start1 = time.time()
        d,p1,p2,count = closest_pair_DnC(arr_points,points,dimensi,count)
        end1 = time.time()
        timed1 = round((end1-start1)*1000,2)
        print("=====")
        print("          DIVIDE AND CONQUER          ")
        print("=====")
        print("Jarak terdekat antara 2 titik : ",round(d,2))
        print("Titik pertama : ",np.round_(p1, decimals=2))
        print("Titik kedua : ",np.round_(p2, decimals=2))
        print("Jumlah operasi euclidean DnC : ",count)
        print("Waktu eksekusi : ",timed1, "ms")
```

```

print("Run in",platform.processor(),"Processor")
print(" ")

count =0
start2 = time.time()
min,p1,p2,count = closest_pair_bf(arr_points,points,dimensi,count)
end2 = time.time()
timed2 = round((end2-start2)*1000,2)
print("=====")
print("                BRUTE FORCE                ")
print("=====")
print("Jarak terdekat antara 2 titik : ",round(d,2))
print("Titik pertama : ",np.round_(p1, decimals=2))
print("Titik kedua : ",np.round_(p2, decimals=2))
print("Jumlah operasi euclidean BF : ",count)
print("Waktu eksekusi : ",timed2, "ms")
print("Run in",platform.processor(),"Processor")

visualisasi(arr_points,p1,p2)

elif(dimensi > 0 and dimensi != 3):
    count = 0
    start1 = time.time()
    d,p1,p2,count = closest_pair_DnC(arr_points,points,dimensi,count)
    end1 = time.time()
    timed1 = round((end1-start1)*1000,2)
    print("=====")
    print("                DIVIDE AND CONQUER                ")
    print("=====")
    print("Jarak terdekat antara 2 titik : ",round(d,2))
    print("Titik pertama : ",np.round_(p1, decimals=2))
    print("Titik kedua : ",np.round_(p2, decimals=2))
    print("Jumlah operasi euclidean DnC : ",count)
    print("Waktu eksekusi : ",timed1, "ms")
    print("Run in",platform.processor(),"Processor")
    print(" ")

    count =0
    start2 = time.time()

```

```

min,p1,p2,count = closest_pair_bf(arr_points,points,dimensi,count)
end2 = time.time()
timed2 = round((end2-start2)*1000,2)
print("=====")
print("                BRUTE FORCE                ")
print("=====")
print("Jarak terdekat antara 2 titik : ",round(d,2))
print("Titik pertama : ",np.round_(p1, decimals=2))
print("Titik kedua : ",np.round_(p2, decimals=2))
print("Jumlah operasi euclidean BF : ",count)
print("Waktu eksekusi : ",timed2, "ms")
print("Run in",platform.processor(),"Processor")

# visualisasi(arr_points,p1,p2)
else:
    print("Dimensi tidak valid")
else:
    print("Jumlah titik atau dimensi tidak valid")

```

3.2 divideandconquer.py

Pada *file* ini terdapat implementasi dari algoritma *Divide and Conquer* yang diawali dengan men-*generate* titik secara *random* sejumlah masukan dari pengguna. Kemudian, melakukan *sorting* terhadap titik yang telah di *generate*. Lalu, terdapat juga implementasi dari perhitungan rumus *euclidean*, yang mana pada akhirnya, melalui pemanfaatan *library* python dan implementasi algoritma tersebut akan didapatkan hasil akhir berupa pasangan titik dengan jarak terdekat.

```

import random
import math

# Generate random titik
def random_points(points,dimensi):
    arr_points = []
    for i in range(points):
        point1 = []
        for j in range(dimensi):
            point2 = random.uniform(-1000,1000)
            point1.append(point2)

```



```

    arr_points.append(point1)
return arr_points

# Function to find the partition position
def partition(arr, low, high):
    pivot = arr[(low+high)//2][0]
    i = low
    j = high
    while True:
        while(arr[i][0] < pivot):
            i = i + 1
        while(arr[j][0] > pivot):
            j = j - 1
        if i < j :
            temp = arr[i]
            arr[i] = arr[j]
            arr[j] = temp
            i = i+1
            j = j-1
        if i>=j :
            break
    return j

# function to perform quicksort
def quickSort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi + 1, high)

# def sortByX(npoint):
#     npoint_sorted = quick_sort(npoint)
#     return closest_pair(npoint_sorted)

def euclidean(x,y,n):
    if n == 1:
        return (x[0]-y[0])
    else:
        return math.sqrt(euclidean(x,y,n-1)**2 + (x[n-1]-y[n-1])**2)

```

```

def count_euclidean(x,y,n,ctr) :
    ctr += 1
    return euclidean(x,y,n), ctr

def closest_pair_DnC(arr,n,dimensi,count):
    if (n == 2):
        d = euclidean(arr[0],arr[1],dimensi)
        count += 1
        return d,arr[0],arr[1],count
    elif (n == 3):
        d1 = euclidean(arr[0],arr[1],dimensi)
        d2 = euclidean(arr[0],arr[2],dimensi)
        d3 = euclidean(arr[1],arr[2],dimensi)
        count += 3
        if (d1 < d2 and d1 < d3):
            return d1,arr[0],arr[1],count
        elif (d2 < d1 and d2 < d3):
            return d2,arr[0],arr[2],count
        else:
            return d3,arr[1],arr[2],count
    else:
        mid = n // 2
        kiri = arr[:mid]
        kanan = arr[mid:]
        d1,p11,p12,count1 = closest_pair_DnC(kiri,mid,dimensi,0)
        d2,p21,p22,count2 = closest_pair_DnC(kanan,n-mid,dimensi,0)
        count += count1 + count2
        d = 0
        p1 = []
        p2 = []
        if (d1 < d2):
            d = d1
            p1 = p11
            p2 = p12
        else:
            d = d2
            p1 = p21
            p2 = p22

```

```

temp = []

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

for i in kiri:
    if i[0] >= mid-d:
        temp.append(i)
for i in kanan:
    if i[0] <= mid+d:
        temp.append(i)

for i in range(len(temp)):
    for j in range(i+1,len(temp)):
        count+=1
        if euclidean(temp[i],temp[j],dimensi) < d:
            d = euclidean(temp[i],temp[j],dimensi)
            p1 = temp[i]
            p2 = temp[j]
return d,p1,p2,count

```

3.3 bruteforce.py

File ini berisi implementasi dari algoritma brute force sebagai pembanding dengan algoritma divide and conquer.

```

from divideandconquer import *

def closest_pair_bf(arr_points,points,dimensi,count):
    min = euclidean(arr_points[0], arr_points[1], dimensi)
    p1 = arr_points[0]
    p2 = arr_points[1]

    for i in range(points):
        for j in range(i+1,points):
            d = euclidean(arr_points[i],arr_points[j],dimensi)
            count+=1

```

```
if d < min:
    min = d
    p1 = arr_points[i]
    p2 = arr_points[j]
return min,p1,p2,count
```

3.4 visualisasi.py

File ini berguna untuk melakukan visualisasi 3D pada titik, sesuai dengan spesifikasi permintaan tugas kecil 2 ini.

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

def visualisasi(arr_points,p1,p2):
    arrX = []
    arrY = []
    arrZ = []

    for i in range(len(arr_points)):
        if (arr_points[i] != p1 and arr_points[i] != p2):
            arrX.append(arr_points[i][0])
            arrY.append(arr_points[i][1])
            arrZ.append(arr_points[i][2])

    fig = plt.figure(figsize=(6,6))
    ax = plt.axes(projection = "3d")

    ax.scatter3D(arrX, arrY, arrZ, color = "blue")
    ax.scatter3D(p1[0], p1[1], p1[2], color="red")
    ax.scatter3D(p2[0], p2[1], p2[2], color="red")

    plt.title("VISUALISASI 3D")
    ax.set_xlabel('X-axis', fontweight = 'bold')
    ax.set_ylabel('Y-axis', fontweight = 'bold')
    ax.set_zlabel('Z-axis', fontweight = 'bold')

    ax.plot([p1[0],p2[0]],[p1[1],p2[1]],[p1[2],p2[2]], c="red")
```

```
# show plot  
plt.show()
```

BAB IV EKSPERIMEN

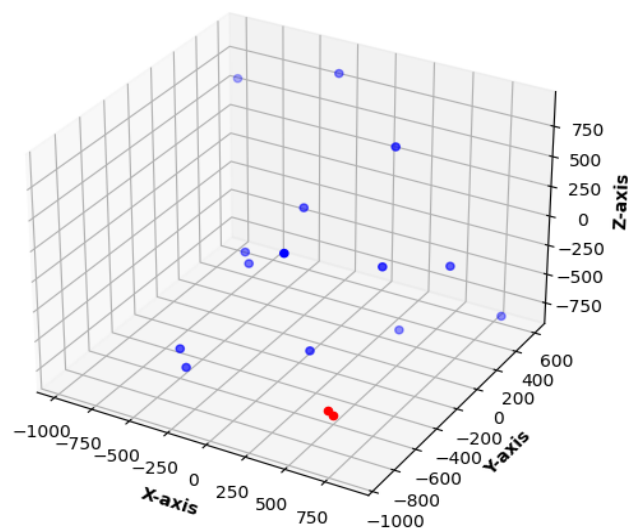
4.1 Points = 16

4.1.1 Dimensi 3

```
Masukkan banyak titik : 16
Masukkan dimensi : 3
=====
DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 43.71
Titik pertama : [ 663.12 -887.07 -501.28]
Titik kedua : [ 699.12 -891.02 -525.77]
Jumlah operasi euclidean DnC : 59
Waktu eksekusi : 0.9987354278564453 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
=====
BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 43.71
Titik pertama : [ 663.12 -887.07 -501.28]
Titik kedua : [ 699.12 -891.02 -525.77]
Jumlah operasi euclidean BF : 120
Waktu eksekusi : 2.9976367950439453 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.1 Luaran untuk points=16 dengan dimensi 3

VISUALISASI 3D



Gambar 4.1 Visualisasi Dimensi 3 dengan points = 16

4.1.2 Dimensi = 2

```
Masukkan banyak titik : 16
Masukkan dimensi : 2
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 87.19
Titik pertama : [ 739.64 -570.32]
Titik kedua : [ 702.73 -649.3 ]
Jumlah operasi euclidean DnC : 39
Waktu eksekusi : 0.9918212890625 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 87.19
Titik pertama : [ 739.64 -570.32]
Titik kedua : [ 702.73 -649.3 ]
Jumlah operasi euclidean BF : 120
Waktu eksekusi : 1.0073184967041016 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.3 Luaran untuk points = 16 dengan dimensi 2

4.1.3 Dimensi = 7

```
Masukkan banyak titik : 16
Masukkan dimensi : 7
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 701.97
Titik pertama : [-409.2  909.89 -772.84 -19.81  301.01 -368.85 -158.95]
Titik kedua : [-109.53  558.97 -842.46 -151.71  802.37 -331.66 -89.47]
Jumlah operasi euclidean DnC : 166
Waktu eksekusi : 4.325151443481445 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 701.97
Titik pertama : [-409.2  909.89 -772.84 -19.81  301.01 -368.85 -158.95]
Titik kedua : [-109.53  558.97 -842.46 -151.71  802.37 -331.66 -89.47]
Jumlah operasi euclidean BF : 120
Waktu eksekusi : 10.999679565429688 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.4 Luaran untuk points = 16 dengan dimensi 7

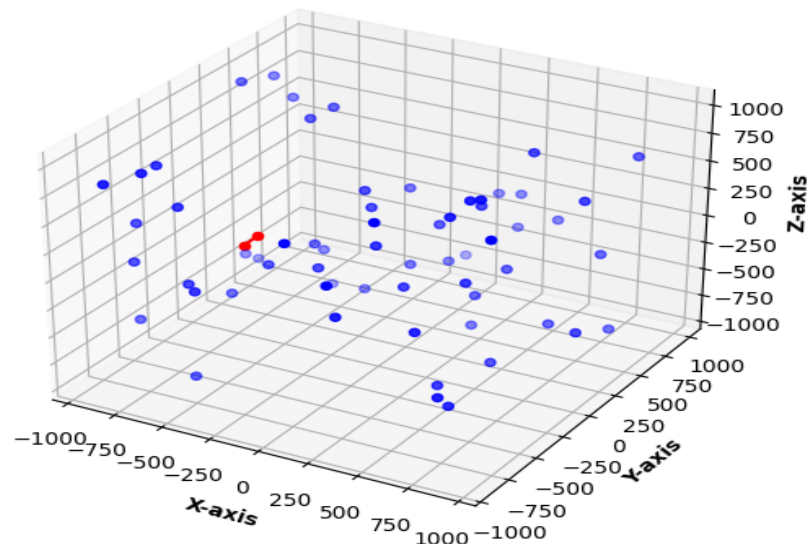
4.2 Points = 64

4.2.1 Dimensi = 3

```
Masukkan banyak titik : 64
Masukkan dimensi : 3
=====
DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 94.49
Titik pertama : [-652.86 -85.06 -141.06]
Titik kedua : [-611.41 -34.22 -73.06]
Jumlah operasi euclidean DnC : 613
Waktu eksekusi : 7.004499435424805 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
=====
BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 94.49
Titik pertama : [-652.86 -85.06 -141.06]
Titik kedua : [-611.41 -34.22 -73.06]
Jumlah operasi euclidean BF : 2016
Waktu eksekusi : 29.946088790893555 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.5 Luaran untuk points = 64 dengan dimensi 3

VISUALISASI 3D



Gambar 4.6 Visualisasi 3D untuk points = 64

4.2.2 Dimensi = 1

```
Masukkan banyak titik : 64
Masukkan dimensi : 1
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : -806.27
Titik pertama : [-105.61]
Titik kedua : [700.66]
Jumlah operasi euclidean DnC : 32
Waktu eksekusi : 0.9992122650146484 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : -806.27
Titik pertama : [-999.79]
Titik kedua : [992.22]
Jumlah operasi euclidean BF : 2016
Waktu eksekusi : 1.9414424896240234 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.7 Luaran untuk points = 64 dengan dimensi 1

4.2.3 Dimensi = 5

```
Masukkan banyak titik : 64
Masukkan dimensi : 5
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 148.95
Titik pertama : [ -37.19 -556.71 493.68 -131.5 757.37]
Titik kedua : [ -86.07 -539.37 565.81 -59.37 662.04]
Jumlah operasi euclidean DnC : 1249
Waktu eksekusi : 35.2628231048584 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 148.95
Titik pertama : [ -37.19 -556.71 493.68 -131.5 757.37]
Titik kedua : [ -86.07 -539.37 565.81 -59.37 662.04]
Jumlah operasi euclidean BF : 2016
Waktu eksekusi : 37.004947662353516 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.8 Luaran untuk points = 64 dengan dimensi 5

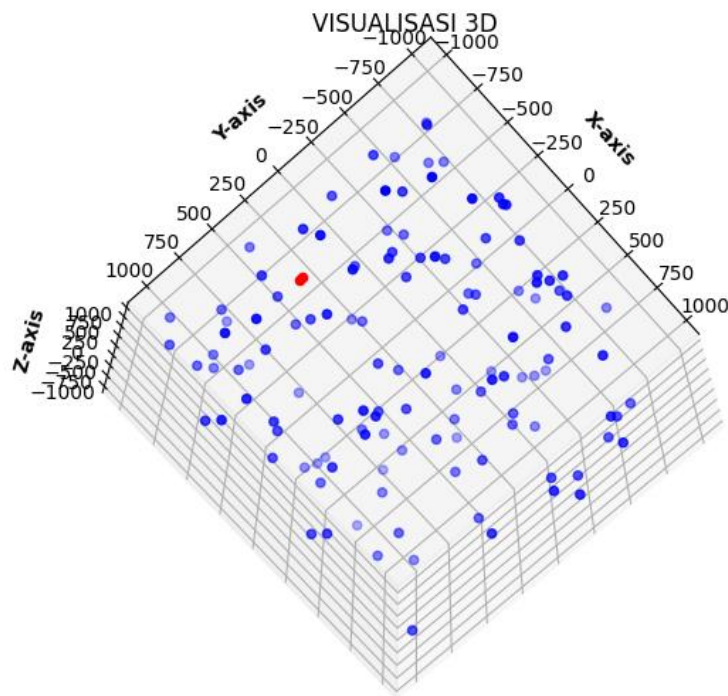
4.3 Points = 128

4.3.1 Dimensi = 3

```
Masukkan banyak titik : 128
Masukkan dimensi : 3
=====
DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 20.64
Titik pertama : [-816.63  71.69 -511.3 ]
Titik kedua : [-815.28  56.28 -497.63]
Jumlah operasi euclidean DnC : 1418
Waktu eksekusi : 18.02849769592285 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 20.64
Titik pertama : [-816.63  71.69 -511.3 ]
Titik kedua : [-815.28  56.28 -497.63]
Jumlah operasi euclidean BF : 8128
Waktu eksekusi : 71.4254379272461 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.9 Luaran untuk points = 128 dengan dimensi 3



Gambar 4.10 Visualisasi 3D untuk points = 128

4.3.2 Dimensi = 2

```
Masukkan banyak titik : 128
Masukkan dimensi : 2

=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 12.54
Titik pertama : [ 386.23 -650.42]
Titik kedua : [ 398.76 -650.68]
Jumlah operasi euclidean DnC : 756
Waktu eksekusi : 8.401632308959961 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 12.54
Titik pertama : [ 11.54 648.51]
Titik kedua : [ 6.84 642.21]
Jumlah operasi euclidean BF : 8128
Waktu eksekusi : 11.922836303710938 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.11 Luaran untuk points = 128 dengan dimensi 2

4.3.3 Dimensi = 4

```
Masukkan banyak titik : 128
Masukkan dimensi : 4

=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 190.86
Titik pertama : [-794.32 138.07 6.73 -769.49]
Titik kedua : [-664.59 20.96 34.12 -697.85]
Jumlah operasi euclidean DnC : 4279
Waktu eksekusi : 35.787105560302734 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 190.86
Titik pertama : [-794.32 138.07 6.73 -769.49]
Titik kedua : [-664.59 20.96 34.12 -697.85]
Jumlah operasi euclidean BF : 8128
Waktu eksekusi : 45.519113540649414 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

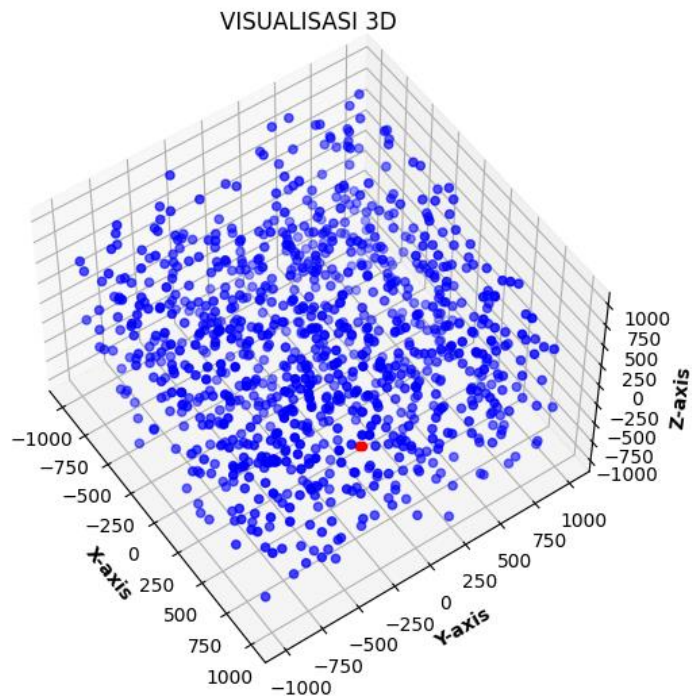
Gambar 4.12 Luaran untuk points = 128 dengan dimensi 4

4.4 Points = 1000

4.4.1 Dimensi = 3

```
Masukkan banyak titik : 1000
Masukkan dimensi : 3
=====
DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 14.29
Titik pertama : [ 953.4 -377.34 572.27]
Titik kedua : [ 959.76 -370.09 561.72]
Jumlah operasi euclidean DnC : 43430
Waktu eksekusi : 236.95039749145508 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
=====
BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 14.29
Titik pertama : [ 953.4 -377.34 572.27]
Titik kedua : [ 959.76 -370.09 561.72]
Jumlah operasi euclidean BF : 499500
Waktu eksekusi : 891.3993835449219 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.13 Luaran untuk points = 1000 dengan dimensi 3



Gambar 4.14 Visualisasi 3D untuk points = 1000

4.4.2 Dimensi = 6

```
Masukkan banyak titik : 1000
Masukkan dimensi : 6
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 151.63
Titik pertama : [ 634.96 -847.06 442.49 -845.5 -224.56 -10.07]
Titik kedua : [ 651.45 -808.36 426.76 -881.32 -355.05 -61.68]
Jumlah operasi euclidean DnC : 221247
Waktu eksekusi : 884.8772048950195 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 151.63
Titik pertama : [ 634.96 -847.06 442.49 -845.5 -224.56 -10.07]
Titik kedua : [ 651.45 -808.36 426.76 -881.32 -355.05 -61.68]
Jumlah operasi euclidean BF : 499500
Waktu eksekusi : 1977.6458740234375 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.15 Luaran untuk points = 1000 dengan dimensi 6

4.4.3 Dimensi = 8

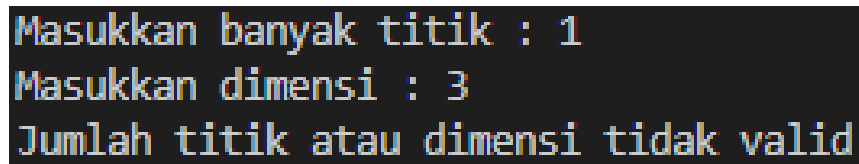
```
Masukkan banyak titik : 1000
Masukkan dimensi : 8
=====
                DIVIDE AND CONQUER
=====
Jarak terdekat antara 2 titik : 358.31
Titik pertama : [-275.64 -909.27 -795.61 213.82 88.3 466.11 441.23 -638.47]
Titik kedua : [-127.07 -845.02 -919.38 338.93 -28.31 642.49 284.11 -595.85]
Jumlah operasi euclidean DnC : 461432
Waktu eksekusi : 2544.505834579468 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor

=====
                BRUTE FORCE
=====
Jarak terdekat antara 2 titik : 358.31
Titik pertama : [-275.64 -909.27 -795.61 213.82 88.3 466.11 441.23 -638.47]
Titik kedua : [-127.07 -845.02 -919.38 338.93 -28.31 642.49 284.11 -595.85]
Jumlah operasi euclidean BF : 499500
Waktu eksekusi : 2596.0676670074463 ms
Run in Intel64 Family 6 Model 78 Stepping 3, GenuineIntel Processor
```

Gambar 4.16 Luaran untuk points = 1000 dengan dimensi 8

4.5 Kasus Points atau Dimensi Tidak Valid

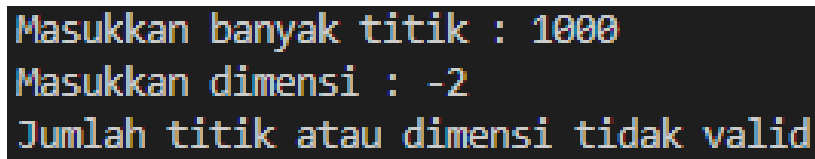
4.5.1 Points = 1



```
Masukkan banyak titik : 1
Masukkan dimensi : 3
Jumlah titik atau dimensi tidak valid
```

Gambar 4.17 Luaran untuk points = 1 dengan dimensi 3

4.5.2 Dimensi = 0



```
Masukkan banyak titik : 1000
Masukkan dimensi : -2
Jumlah titik atau dimensi tidak valid
```

Gambar 4.18 Luaran untuk points = 1000 dengan dimensi 0

LAMPIRAN

Link repository github :

https://github.com/satrianababan/Tucil2_13521168

Tabel check list :

Poin	Ya	Tidak
1. Program berhasil di kompilasi 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	✓	

DAFTAR PUSTAKA

Munir, Rinaldi (2022). *Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Divide and Conquer (Bagian 1)*. Diakses pada 25 Januari 2023 pukul 22.10 dari sumber [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

Munir, Rinaldi (2022). *Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Divide and Conquer (Bagian 2)*. Diakses pada 25 Januari 2023 pukul 23.15 dari sumber [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf)