Laporan Tugas Kecil 2

IF2211 - Strategi Algoritma

Semester II Tahun Ajaran 2022/2023

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Disusun oleh:

Muchammad Dimas Sakti Widyatmaja

13521160

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 4013

I. Deskripsi Persoalan

Mencari sepasang titik terdekat dengan Algoritma Divide and Conquer sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini Anda 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 titk P1 = (x1, y1, z1) dan P2 = (x2, y2, z2) dihitung dengan rumus Euclidean berikut:

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

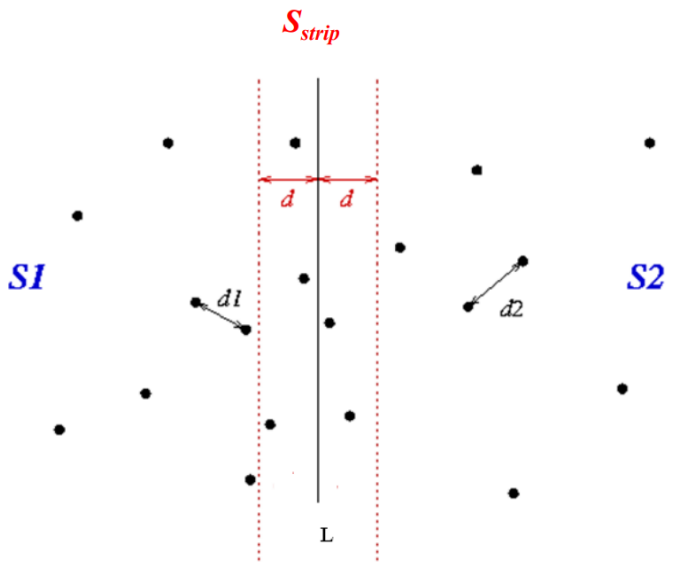
II. Algoritma Closest Pair

1. Brute Force

Algoritma brute force merupakan algoritma yang menggunakan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma brute force merupakan algoritma yang relative sederhana. Karakteristik algoritma Brute Force umumnya tidak mangkus dan sangkil, karena membutuhkan jumlah langkah yang besar dalam penyelesaiannya, sehingga terkadang algoritma Brute Force disebut juga sebagai algoritma yang naif. Hampir semua persoalan dapat diselesaikan dengan algoritma brute force. Termasuk persoalan mencari titik terdekat ini. Algoritma brute force berjalan yang saya gunakan untuk memecahkan persoalan mencari dua titik terdekat adalah sebagai berikut.

1) Buat variabel untuk menyimpan jarak minimum dan inisiasi nilainya
2) Buat nested loop dengan dua tingkat untuk mengiterasi semua kombinasi antara titik
3) Pada tiap iterasi, hitung jarak euclidean antara dua titik yang dicari pada kalang
4) Bandingkan jarak tersebut dengan variable jarak minimum, apabila lebih kecil maka ganti isi variabel jarak minimum dengan jarak tersebut.

2. Divide and Conquer



Keterangan: d = MIN($d1$, $d2$)

Algoritma Divide and Conquer pada umumnya terdiri atas tiga bagian utama yaitu divide, conquer, dan combine. Divide membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Conquer (solve) menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara

rekursif jika masih berukuran besar). Combine mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula. Implementasi dari algortima divide and conquer untuk mencari pasangan titik terdekat adalah sebagai berikut.

1) Jika n = 2, maka jarak kedua titik dihitung langsung dengan rumus Euclidean. Jika n = 3, maka cari jarak terdekat di antara ketiga titik tersebut dengan kondisional.
2) 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 = x[n/2].
3) Secara rekursif, terapkan algoritma D-and-C pada masing-masing bagian untuk mencari sepasang titik terdekat.
4) Pasangan titik yang jaraknya terdekat ada tiga kemungkinan letaknya: Pasangan titik terdekat terdapat di dalam bagian S1. Pasangan titik terdekat terdapat di dalam bagian S2. Pasangan titik terdekat dipisahkan oleh garis batas L, yaitu satu titik di S1 dan satu titik di S2.
5) Jika terdapat pasangan titik p_{left} and p_{right} yang jaraknya lebih kecil dari d, maka kasusnya adalah: Absis x dari p_{left} dan p_{right} berbeda paling banyak sebesar d. Ordinat y dari p_{left} dan p_{right} berbeda paling banyak sebesar d.
6) Temukan semua titik di S1 yang memiliki absis x minimal x[n/2] – d dan absis x maksimal x[n/2] + d lalu masukkan himpunan titik-titik tersebut ke dalam senarai S_{strip}.
7) Urutkan titik-titik di dalam S_{strip} dalam urutan ordinat y yang menaik. Hitung jarak setiap pasang titik di dalam S_{strip} dan bandingkan apakah jaraknya lebih kecil dari d.

III. Source Code

closestDistance.py

```
from typing import List
from quickSort import *
import math

def calcDistance(p1, p2):
    global timesDistanceCalculated
    # Calculate Euclidean distance between two points

    timesDistanceCalculated += 1
    sum = 0
    for i in range(len(p1)):
        sum += (p1[i] - p2[i])**2
    return math.sqrt(sum)

def calcDistance3(P):
    # Calculate Euclidean distance between three points

    if (calcDistance(P[0], P[1]) < calcDistance(P[0], P[2])):
        if (calcDistance(P[0], P[1]) < calcDistance(P[1], P[2])):
            return calcDistance(P[0], P[1]), P[0], P[1]
        else:
            return calcDistance(P[1], P[2]), P[1], P[2]
    else:
        if (calcDistance(P[0], P[2]) < calcDistance(P[1], P[2])):
            return calcDistance(P[0], P[2]), P[0], P[2]
        else:
            return calcDistance(P[1], P[2]), P[1], P[2]

def findClosestPair(P: List, n: int):
    # sort the list first based on x
    quicksort(P, 0, n-1)

    # Base case when there are only two or three points
    if n == 2:
        d = calcDistance(P[0], P[1])
        return d, P[0], P[1]

    if n == 3:
        return calcDistance3(P)

    # Divide the set into two halves
```

```
    mid = n // 2
    S1 = P[:(mid+(n%2))]
    S2 = P[(mid+(n%2)):]

    # Recursive calls to find the closest pair in each half
    d1, p1, p2 = findClosestPair(S1, mid+(n%2))
    d2, q1, q2 = findClosestPair(S2, n-(mid+(n%2)))
    d, p1, p2 = (d1, p1, p2) if d1 < d2 else (d2, q1, q2)

    # Find the minimum distance between the two halves
    d = min(d1, d2)

    closest_pair = (d, p1, p2)
    strip = []
    for i in range(n):
        if abs(P[i][0] - P[mid+(n%2)][0]) < d:
            strip.append(P[i])

    strip.sort(key=lambda x: x[1])
    size = len(strip)

    for i in range(size):
        for j in range(i+1, size):
            if strip[j][1] - strip[i][1] >= d:
                continue
            else:
                distance = calcDistance(strip[i],strip[j])
                d = min(d, distance)
                if (d == distance):
                    closest_pair = (d,strip[i],strip[j])

    return closest_pair

def findClosestPairES(P):
    n = len(P)
    best_dist = float('inf')
    best_pair = None
    for i in range(n):
        for j in range(i+1, n):
            dist = calcDistance(P[i], P[j])
            if dist < best_dist:
                best_dist = dist
                best_pair = (P[i], P[j])

    print(f"Jarak terdekat Brute Force: {best_dist:.2f}")
```

```
    print(f"Pasangan titik terdekat Brute Force: {best_pair}")
    # return best_pair, best_dist
```

quickSort.py

```
def partition(arr, left, right):
    pivot = arr[right][0]
    i = left - 1
    for j in range(left, right):
        if arr[j][0] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i+1], arr[right] = arr[right], arr[i+1]
    return i+1

def quicksort(arr, left, right):
    if left < right:
        id = partition(arr, left, right)
        quicksort(arr, left, id-1)
        quicksort(arr, id+1, right)
```

IO.py

```
import random
import matplotlib.pyplot as plt

def inputRandom(count, dimension):
    vectorList = []

    for i in range(count):
        vector = ()
        for j in range(dimension):
            vector += (random.randint(-5000, 5000),)

        vectorList.append(vector)

    # print("Titik-titik hasil input acak: ", vectorList)
    return vectorList

def show3d(vectorList, res1, res2):
    # create 3D object axes
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
```

```
    # Adding points to the plot
    for point in vectorList:
        if point == res1 or point == res2:
            ax.scatter(point[0], point[1], point[2], c = "green") # green marker
used for closest points
        else:
            ax.scatter(point[0], point[1], point[2], c = "red") # red marker used
for other points

    plt.show()
```

main.py

```
from closestDistance import *
from IO import *
import time

if __name__ == "__main__":
    print('CLOSEST DISTANCE')
    print('CLOSEST DISTANCE\n')
    print('DIVIDE AND CONQUER')
    print('DIVIDE AND CONQUER')
    print('=====================================================')
    count = int(input("Masukkan jumlah tuple: "))
    dimension = int(input("Masukkan dimensi vektor: "))
    vectorList = inputRandom(count, dimension)
    print('=====================================================')
    timesDistanceCalculated = 0

    # catat waktu awal dnd
    start_time = time.time()

    d, res1, res2 = findClosestPair(vectorList, count)

    # catat waktu selesai dnd
    end_time = time.time()

    # hitung selisih waktu
    total_time = end_time - start_time

    print(f"Jarak terdekat Divide and Conquer: {d:.2f}")
    print(f"Pasangan titik terdekat Divide and Conquer: {res1}, {res2}")
    print(f"Waktu yang diperlukan Divide and Conquer: {total_time:.8f} detik")
```

```
    print(f"Jumlah perhitungan jarak euclidean yang dilakukan:
{timesDistanceCalculated}")
    print('=====================================================')

    # catat waktu awal bf
    start_time = time.time()

    findClosestPairES(vectorList)

    # catat waktu selesai bf
    end_time = time.time()

    # hitung selisih waktu
    total_time = end_time - start_time

    print(f"Waktu yang diperlukan Brute Force: {total_time:.8f} detik")
    print('=====================================================')

    # visualisasikan titik-titik
    if (dimension == 3):
        show3d(vectorList, res1, res2)
```

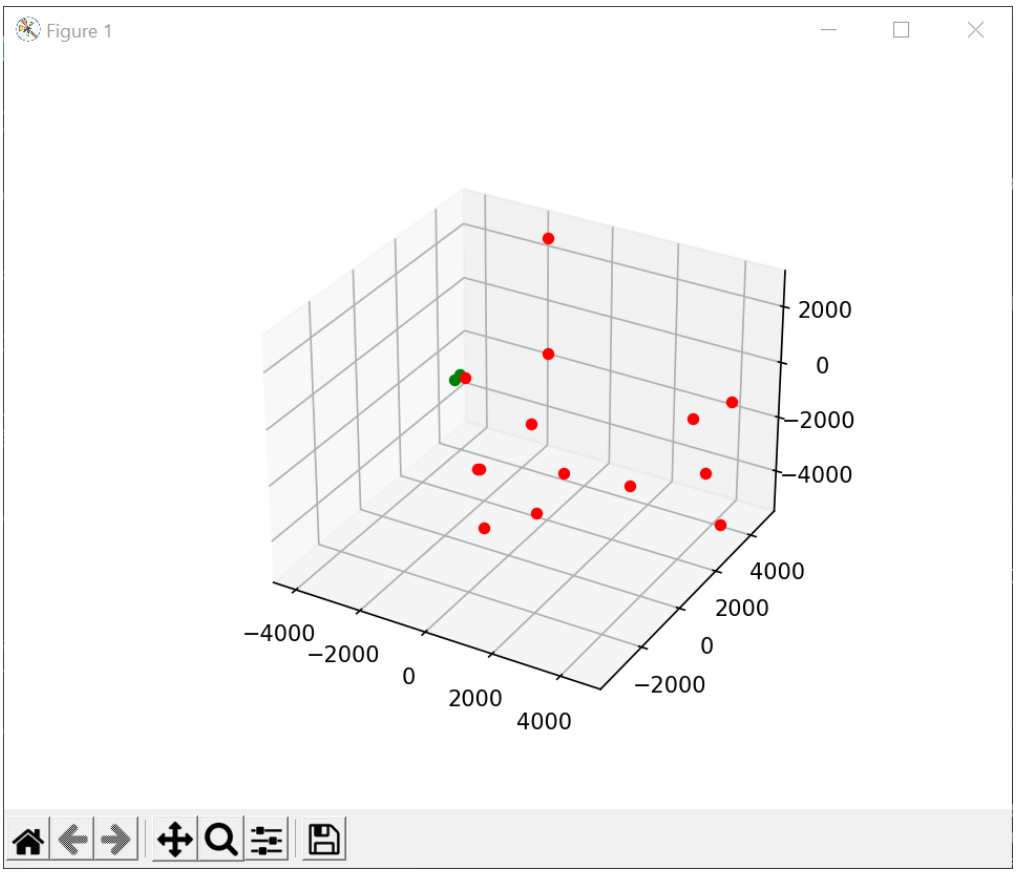
IV. Hasil

Berikut adalah hasil percobaan terhadap beberapa kasus uji yang telah disiapkan. Program dijalankan pada laptop Windows 10 dengan prosesor AMD Ryzen 3 3200U, VGA AMD Radeon RX Vega 3 dan RAM 8 GB.

i. Dimensi 3

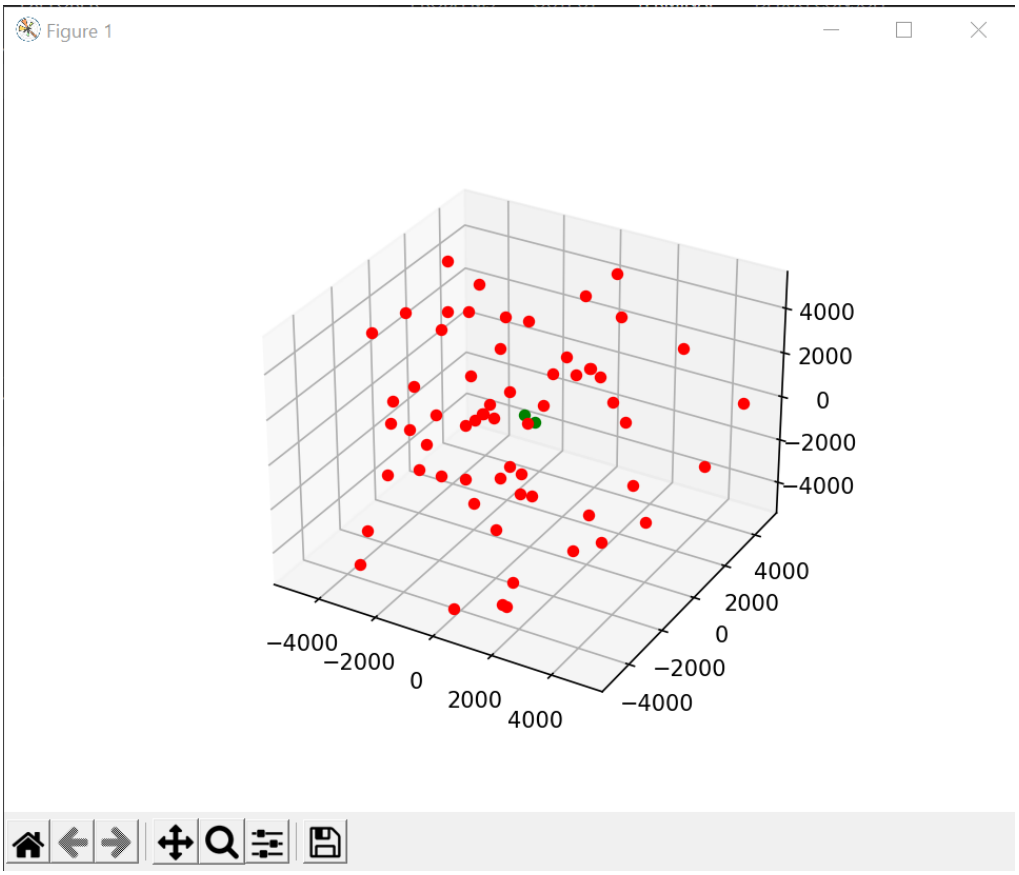
1. n = 16

```
CLOSEST DISTANCE
DIVIDE AND CONQUER
==================================================
Masukkan jumlah tuple: 16
Masukkan dimensi vektor: 3
==================================================
Jarak terdekat Divide and Conquer: 210.25
Pasangan titik terdekat Divide and Conquer: (-4080, 3815, -2733), (-3980, 3937, -2594)
Waktu yang diperlukan Divide and Conquer: 0.00100446 detik
Jumlah perhitungan jarak eucledian yang dilakukan: 26
==================================================
Jarak terdekat Brute Force: 210.25
Pasangan titik terdekat Brute Force: ((-4080, 3815, -2733), (-3980, 3937, -2594))
Waktu yang diperlukan Brute Force: 0.00113320 detik
==================================================
```



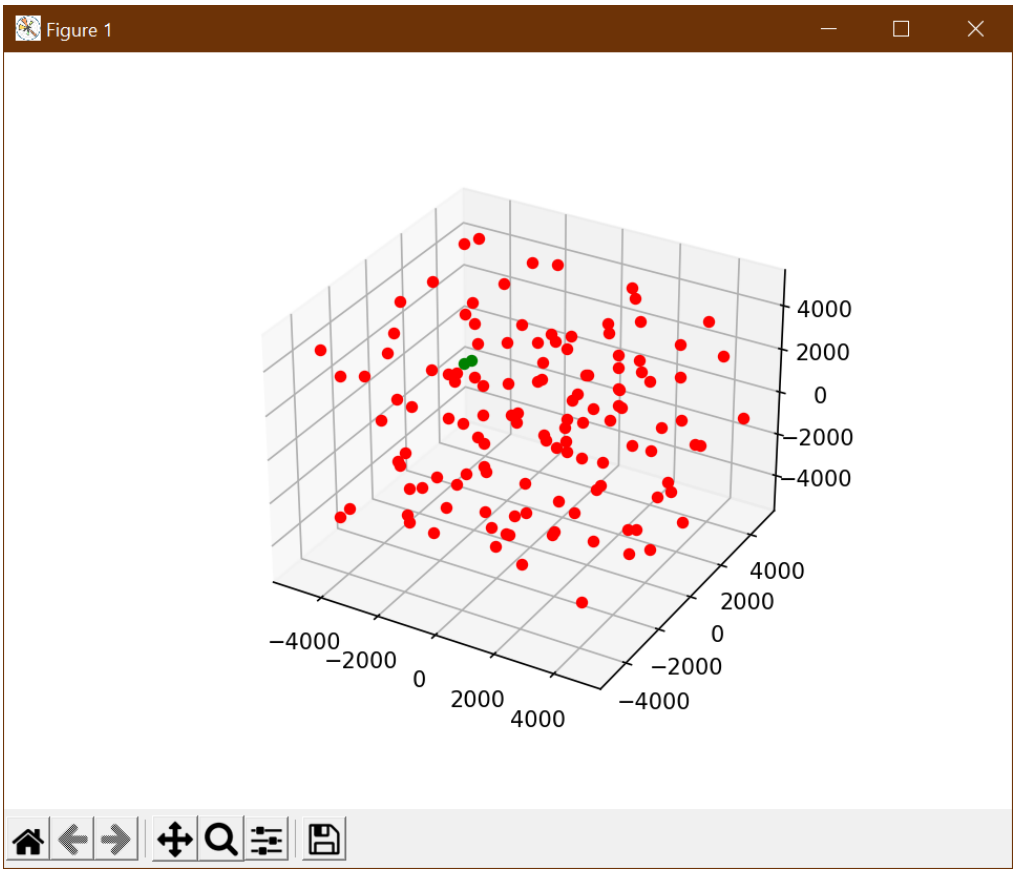
2. n = 64

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗ ██╗███████╗████████╗ █████╗ ███╗   ██╗ ██████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██║██╔════╝╚══██╔══╝██╔══██╗████╗  ██║██╔════╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██║  ██║██║███████╗   ██║   ███████║██╔██╗ ██║██║     █████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██║  ██║██║╚════██║   ██║   ██╔══██║██║╚██╗██║██║     ██╔══╝
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██████╔╝██║███████║   ██║   ██║  ██║██║ ╚████║╚██████╗███████╗
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═════╝ ╚═╝╚══════╝   ╚═╝   ╚═╝  ╚═╝╚═╝  ╚═══╝ ╚═════╝╚══════╝

 ██████╗ ██╗██╗   ██╗██╗██████╗ ███████╗     █████╗ ███╗   ██╗██████╗      ██████╗ ██████╗ ███╗   ██╗ ██████╗ ██╗   ██╗███████╗██████╗
 ██╔══██╗██║██║   ██║██║██╔══██╗██╔════╝    ██╔══██╗████╗  ██║██╔══██╗    ██╔════╝██╔═══██╗████╗  ██║██╔═══██╗██║   ██║██╔════╝██╔══██╗
 ██║  ██║██║██║   ██║██║██║  ██║█████╗      ███████║██╔██╗ ██║██║  ██║    ██║     ██║   ██║██╔██╗ ██║██║   ██║██║   ██║█████╗  ██████╔╝
 ██║  ██║██║╚██╗ ██╔╝██║██║  ██║██╔══╝      ██╔══██║██║╚██╗██║██║  ██║    ██║     ██║   ██║██║╚██╗██║██║▄▄ ██║██║   ██║██╔══╝  ██╔══██╗
 ██████╔╝██║ ╚████╔╝ ██║██████╔╝███████╗    ██║  ██║██║ ╚████║██████╔╝    ╚██████╗╚██████╔╝██║ ╚████║╚██████╔╝╚██████╔╝███████╗██║  ██║
 ╚═════╝ ╚═╝  ╚═══╝  ╚═╝╚═════╝ ╚══════╝    ╚═╝  ╚═╝╚═╝  ╚═══╝╚═════╝      ╚═════╝ ╚═════╝ ╚═╝  ╚═══╝ ╚══▀▀═╝  ╚═════╝ ╚══════╝╚═╝  ╚═╝

==================================================
Masukkan jumlah tuple: 64
Masukkan dimensi vektor: 3
==================================================
Jarak terdekat Divide and Conquer: 443.47
Pasangan titik terdekat Divide and Conquer: (-641, 1372, -817), (-1060, 1446, -692)
Waktu yang diperlukan Divide and Conquer: 0.00299716 detik
Jumlah perhitungan jarak eucledian yang dilakukan: 204
==================================================
Jarak terdekat Brute Force: 443.47
Pasangan titik terdekat Brute Force: ((-1060, 1446, -692), (-641, 1372, -817))
Waktu yang diperlukan Brute Force: 0.00657630 detik
==================================================
```



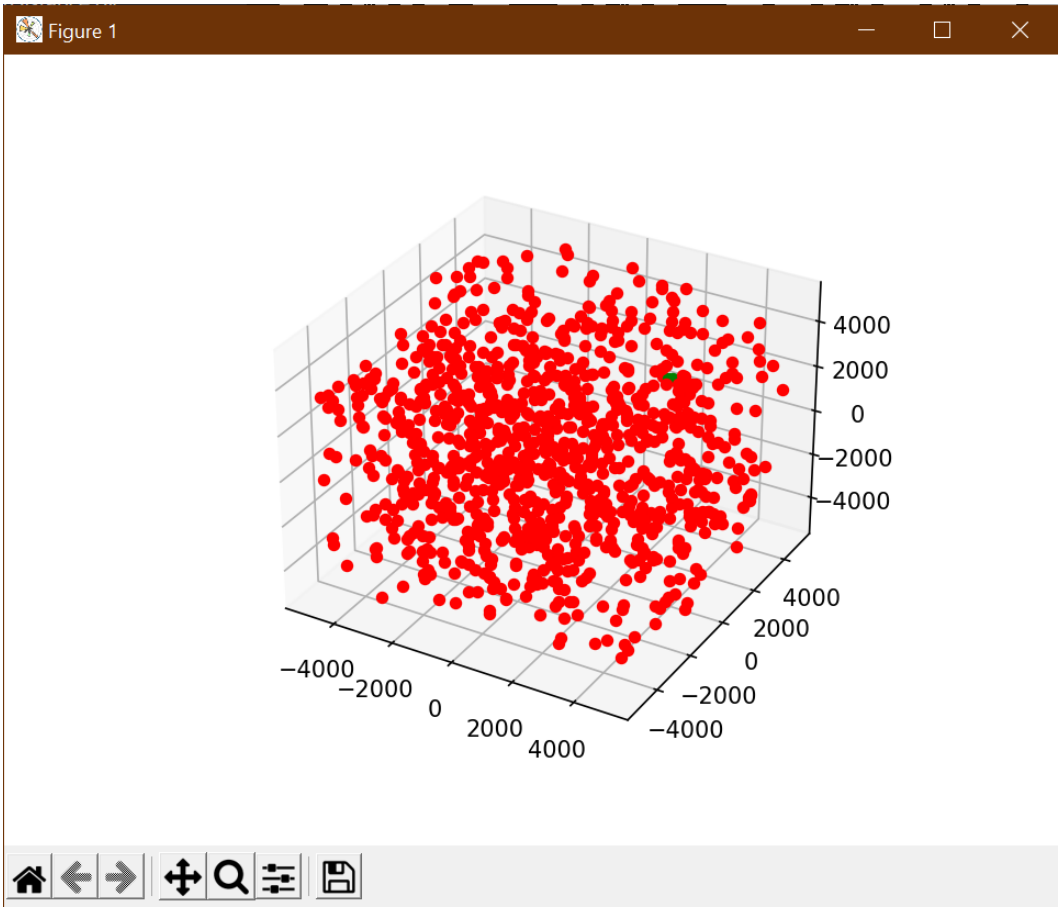
3. n = 128

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗ ██╗███████╗████████╗ █████╗ ███╗   ██╗ ██████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██║██╔════╝╚══██╔══╝██╔══██╗████╗  ██║██╔════╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██║  ██║██║███████╗   ██║   ███████║██╔██╗ ██║██║     █████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██║  ██║██║╚════██║   ██║   ██╔══██║██║╚██╗██║██║     ██╔══╝
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██████╔╝██║███████║   ██║   ██║  ██║██║ ╚████║╚██████╗███████╗
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═════╝ ╚═╝╚══════╝   ╚═╝   ╚═╝  ╚═╝╚═╝  ╚═══╝ ╚═════╝╚══════╝

 ██████╗ ██╗██╗   ██╗██╗██████╗ ███████╗     █████╗ ███╗   ██╗██████╗      ██████╗ ██████╗ ███╗   ██╗ ██████╗ ██╗   ██╗███████╗██████╗
 ██╔══██╗██║██║   ██║██║██╔══██╗██╔════╝    ██╔══██╗████╗  ██║██╔══██╗    ██╔════╝██╔═══██╗████╗  ██║██╔═══██╗██║   ██║██╔════╝██╔══██╗
 ██║  ██║██║██║   ██║██║██║  ██║█████╗      ███████║██╔██╗ ██║██║  ██║    ██║     ██║   ██║██╔██╗ ██║██║   ██║██║   ██║█████╗  ██████╔╝
 ██║  ██║██║╚██╗ ██╔╝██║██║  ██║██╔══╝      ██╔══██║██║╚██╗██║██║  ██║    ██║     ██║   ██║██║╚██╗██║██║▄▄ ██║██║   ██║██╔══╝  ██╔══██╗
 ██████╔╝██║ ╚████╔╝ ██║██████╔╝███████╗    ██║  ██║██║ ╚████║██████╔╝    ╚██████╗╚██████╔╝██║ ╚████║╚██████╔╝╚██████╔╝███████╗██║  ██║
 ╚═════╝ ╚═╝  ╚═══╝  ╚═╝╚═════╝ ╚══════╝    ╚═╝  ╚═╝╚═╝  ╚═══╝╚═════╝      ╚═════╝ ╚═════╝ ╚═╝  ╚═══╝ ╚══▀▀═╝  ╚═════╝ ╚══════╝╚═╝  ╚═╝

==========================================================
Masukkan jumlah tuple: 128
Masukkan dimensi vektor: 3
==========================================================
Jarak terdekat Divide and Conquer: 258.28
Pasangan titik terdekat Divide and Conquer: (-2440, 222, 1866), (-2270, 375, 1986)
Waktu yang diperlukan Divide and Conquer: 0.00863338 detik
Jumlah perhitungan jarak euclidean yang dilakukan: 431
==========================================================
Jarak terdekat Brute Force: 258.28
Pasangan titik terdekat Brute Force: ((-2440, 222, 1866), (-2270, 375, 1986))
Waktu yang diperlukan Brute Force: 0.02502227 detik
==========================================================
```



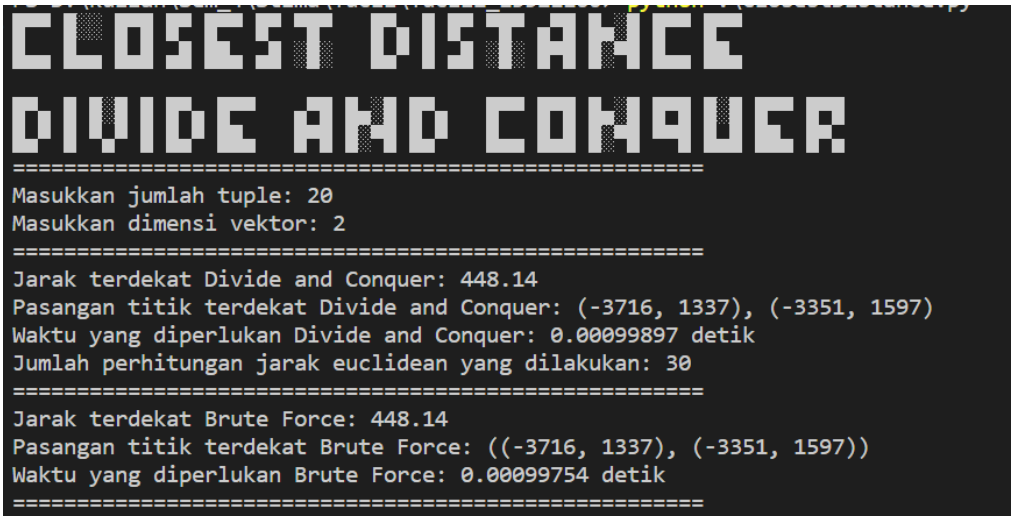
4. n = 1000

```
CLOSEST DISTANCE
DIVIDE AND CONQUER
=====================================================
Masukkan jumlah tuple: 1000
Masukkan dimensi vektor: 3
=====================================================
Jarak terdekat Divide and Conquer: 109.22
Pasangan titik terdekat Divide and Conquer: (1869, 3680, 1310), (1969, 3718, 1332)
Waktu yang diperlukan Divide and Conquer: 0.36217809 detik
Jumlah perhitungan jarak euclidean yang dilakukan: 4504
=====================================================
Jarak terdekat Brute Force: 109.22
Pasangan titik terdekat Brute Force: ((1869, 3680, 1310), (1969, 3718, 1332))
Waktu yang diperlukan Brute Force: 1.50804043 detik
=====================================================
```

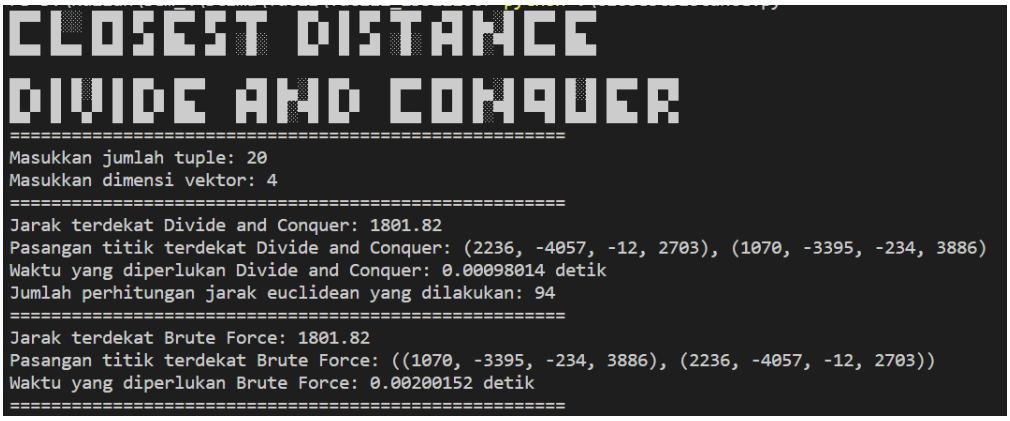


ii. Dimensi general

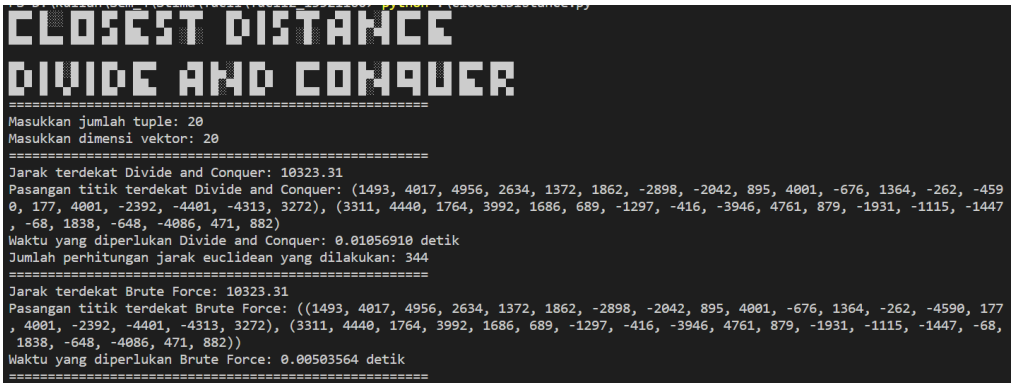
1. Dimensi 2 dan n = 20



2. Dimensi 4 dan n = 20



3. Dimensi 20 dan n = 20



V. Lampiran

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	✓	

Tautan repositori: https://github.com/SaktiWidyatmaja/Tucil2_13521160