Laporan Tugas Kecil II

IF2211 STRATEGI ALGORITMA

Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer*

Disusun oleh:

Kartini Copa 13521026

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung 2023

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

1.1 Deskripsi Masalah

Mencari pasangan 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 pasangan titik terdekat pada bidang 3D. Misalkan terdapat n buah titik pada ruang 3D. Setiap titik P di dalam ruang dinyatakan dengan koordinat $P = (x, y, z)$. Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titik $P_1 = (x_1, y_1, z_1)$ dan $P_2 = (x_2, y_2, z_2)$ dihitung dengan rumus Euclidean berikut:

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

Buatlah program dalam dalam Bahasa C/C++/Java/Python/Golang/Ruby/Perl (pilih salah satu) untuk mencari pasangan titik yang jaraknya terdekat satu sama lain dengan menerapkan algoritma divide and conquer untuk penyelesaiannya, dan perbandingannya dengan Algoritma Brute Force. Masukan program: titik-titik n (dibangkitkan secara acak) dalam koordinat (x, y, z)

Luaran program:

- sepasang titik yang jaraknya terdekat dan nilai jaraknya

- banyaknya operasi perhitungan rumus Euclidean

- 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 pasangan titik terdekat untuk sekumpulan vektor di Rn , setiap vektor dinyatakan dalam bentuk x = (x1, x2, …, xn)

BAB II

Strategi Algoritma

2.1 Algoritma *Brute Force*

Algoritma brute force adalah algoritma mencari semua kemungkinan solusi untuk memecahkan suatu masalah. Namun, algoritma ini biasanya bergantung pada kekuatan komputasi yang tinggi untuk mendapatkan semua solusi yang tepat daripada menggunakan teknik yang canggih sehingga sangat lambat dan tidak efisien untuk masalah atau kasus yang besar. Walaupun algoritma brute force memiliki kelemahan dalam efisiensi, namun algoritma ini memiliki keunggulan dapat menyelesaikan hampir semua permasalahan dengan tepat.

Algoritma *brute force* dapat menyelesaikan permasalahan mencari pasangan titik terdekat pada ruang dimensi tiga. Namun pada jumlah titik yang tidak berhingga, waktu eksekusi sangat lama sehingga algoritma *brute force* kurang efisien dalam menyelesaikan permasalah tersebut. Berikut adalah langkah-langkah untuk mencari pasangan titik terdekat dengan algoritma brute force:

1. Hitung jumlah titik yang ada dalam himpunan titik.
2. Inisialisasi jarak terdekat dengan nilai yang besar atau tak terhingga.
3. Perulangan untuk setiap pasangan titik (titik i dan titik j), dimana $i \neq j$.
4. Hitung jarak euclidean antara titik i dan titik j menggunakan rumus Euclidean
5. Jika jarak distance lebih kecil dari nilai jarak tersekat saat ini, update jarak terdekat dengan jarak.
6. Setelah selesai melakukan perulangan untuk setiap pasangan titik, jarak terdekat akan berisi jarak terpendek dari seluruh pasangan titik.

Algoritma brute force ini sederhana dan mudah dipahami, namun memiliki kompleksitas waktu yang tinggi yaitu $O(n^2)$. Oleh karena itu, algoritma ini lebih cocok digunakan pada himpunan titik yang kecil. Karena untuk setiap titik, perlu memeriksa jarak semua titik lain untuk menemukan pasangan terdekat. Dalam kasus ini, perhitungan sebanyak n kali, sehingga kompleksitas waktu algoritma adalah $O(n^2)$.

2.2 Algoritma *Divide and Conquer*

Algoritma *divide and conquer* merupakan salah satu strategi algoritma yang menyelesaikan persoalan dengan cara membagi persoalan yang besar menjadi persoalan yang memecah masalah menjadi bagian-bagian yang lebih kecil, menyelesaikan setiap bagian secara terpisah, dan kemudian menggabungkan penyelesaian tersebut menjadi satu solusi. Langkah yang dilakukan dalam algoritma *divide and conquer* ada 3 yaitu, *divide, conquer,* dan *combine*.

Divide merupakan tahap membagi persoalan menjadi beberapa upa-masalah yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil. *Conque*r merupakan tahap penyelesaian setiap upa-masalah (secara langsung atau secara rekursif). *Combine* merupakan tahap penggabungan solusi masing-masing upa-masalah sehingga membentuk solusi persoalan utuh.

Algoritma *divide and conquer* dapat digunakan dalam menyelesaikan permasalahan mencari pasangan titik terdekat pada ruang dimensi tiga. Berikut merupakan penjelasan untuk setiap bagian algoritma *divide and conquer*.

1. *Divide*: himpunan titik dibagi menjadi dua upa-himpunan dengan dengan titik tengah. Setiap upa-himpunan diproses secara rekursif dengan algoritma *divide and conquer* hingga ukuran himpunan titik menjadi 1 atau 2.

2. *Conquer*: menyelesaikan upa-masalah terkecil, yaitu mencari jarak terpendek antara dua titik. Jika terdapat hanya satu atau dua titik, jarak terpendek dapat langsung dicari. Jika terdapat lebih dari dua titik, algoritma akan memanggil dirinya sendiri untuk mencari jarak terpendek pada bagian kiri dan kanan.

3. *Combine*: menggabungkan solusi dari upa-masalah untuk mencari jarak terpendek antara dua titik dari kedua upa-*array*. Algoritma mencari jarak terpendek antara dua titik pada bagian tengah himpunan titik. Jarak terpendek yang dihasilkan akan menjadi solusi utuh.

Kompleksitas algoritma *divide and conquer* dalam pencarian titik terdekat pada ruang dimensi tiga dapat dihitung dengan mempertimbangkan dua bagian utama dari algoritma yaitu, pembagian (*divide*) dan penggabungan (*combine*). Bagian pembagian dilakukan dengan membagi himpunan titik menjadi dua upa-himpunan yang sama besarnya secara rekursif. Kompleksitas waktu untuk melakukan pembagian adalah O(1), yaitu konstan. Bagian penggabungan dilakukan dengan mencari pasangan titik terdekat antara pasangan titik terdekat di upa-himpunan kiri dan kanan, kemudian mencari pasangan titik terdekat di antara keduanya. Untuk mencari pasangan titik terdekat antara kedua upa-himpunan, algoritma menggunakan strategi divide and conquer dengan kompleksitas waktu O(n log n). Untuk mencari pasangan titik terdekat di antara kedua upa-himpunan, algoritma melakukan iterasi pada setiap titik di upa-himpunan tengah yang terletak pada jarak kurang dari jarak pasangan titik terdekat yang ditemukan pada langkah sebelumnya. Kompleksitas waktu untuk mencari pasangan titik terdekat di antara kedua upa-himpunan adalah O(n). Dengan demikian, kompleksitas waktu total algoritma divide and conquer pada kode di atas adalah O(n log n).

BAB III

Implementasi dan Pengujian

3.1 Implementasi Program

Program pencarian titik terdekat dengan implementasi algoritma *divide and conquer* dibuat dalam bahasa pemrograman Python. Program ini menggunakan beberapa *library* sebagai berikut.

1. math: *library* ini berfungsi untuk perhitungan jarak Euclidean
2. random: *library* ini berfungsi untuk membangkitkan titik-titik acak
3. time: *library* ini berfungsi untuk menghitung waktu eksekusi program
4. platform: *library* ini berfungsi untuk informasi spesifikasi komputer
5. matplotlib.pyplot: *library* ini berfungsi untuk menggambarkan titik-titik pada ruang

3.1.1 Implementasi Algoritma *Brute Force*

Berikut merupakan pencarian titik terdekat dengan algoritma brute force sebagai perbandingan yang terdapat dalam *file* brute_force.py.

```python
import math

def eucl_distance(p1, p2):
    return math.sqrt(sum((p1[i] - p2[i])**2 for i in range(len(p1))))

def brute_force(points):
    global eucl_brute
    eucl_brute = 0
    n = len(points)
    if n <= 1:
        return None
    elif n == 2:
        eucl_brute += 1
        return (points[0], points[1], eucl_distance(points[0], points[1]))
    else:
        min_dist = eucl_distance(points[0], points[1])
        p1 = points[0]
        p2 = points[1]
        for i in range(n):
            for j in range(i+1, n):
                distance = eucl_distance(points[i], points[j])
                eucl_brute += 1
                if distance < min_dist:
                    min_dist = distance
                    p1 = points[i]
                    p2 = points[j]
        return (p1, p2, min_dist)
```

Gambar 3.1.1 brute_force.py

3.1.2 Implementasi Algoritma *Divide and Conquer*

Berikut merupakan implementasi algoritma *divide and conquer* dalam pencarian titik terdekat yang terdapat pada *file* closest_pair.py.

```python
import math

eucl_dnc = 0

def eucl_distance(p1, p2):
    return math.sqrt(sum((p1[i] - p2[i])**2 for i in range(len(p1))))

def div_conquer(points):
    global eucl_dnc
    n = len(points)
    if n <= 1:
        return None
    elif n == 2:
        eucl_dnc += 1
        distance = eucl_distance(points[0], points[1])
        return (points[0], points[1], distance)
    else:
        # Sort points berdasarkan kordinat x
        sortPoint = sorted(points, key=lambda x: x[0])
        midPoint = n // 2
        leftPoint = sortPoint[:midPoint]
        rightPoint = sortPoint[midPoint:]
        # Closest pair yang berada di subarray kiri dan kanan
        leftClosest = div_conquer(leftPoint)
        rightClosest = div_conquer(rightPoint)
        # Cari closest pair yang terdekat
        if leftClosest is None and rightClosest is None:
            closest = None
        elif leftClosest is None:
            closest = rightClosest
        elif rightClosest is None:
            closest = leftClosest
        else:
            closest = leftClosest if leftClosest[2] < rightClosest[2] else rightClosest
        # Closest pair yang berada di antara kedua subarray
        # Sort points berdasarkan kordinat y
        closest_mid = [point for point in sortPoint if abs(point[0] - sortPoint[midPoint][0]) < closest[2]]
        closest_mid = sorted(closest_mid, key=lambda x: x[1])
        for i, point1 in enumerate(closest_mid):
            for point2 in closest_mid[i+1:]:
                if point2[1] - point1[1] >= closest[2]:
                    break
                distance = eucl_distance(point1, point2)
                eucl_dnc += 1
                if distance < closest[2]:
                    closest = (point1, point2, distance)
        return closest
```

Gambar 3.1.2 closest_pair.py

3.1.3 Implementasi *Generate Points*

Berikut ini merupakan implementasi dari *generate points* untuk membangkitkan titik-titik secara acak yang terdapat pada file main.py.

```
def generate_points(n, Rn):
    points = []
    for i in range(n):
        point = []
        for j in range(Rn):
            point.append(random.random())
        points.append(point)
    return points
```

Gambar 3.1.3 generate_points

3.1.4 Implementasi Program main.py

```
def splash_screen():
    print("                                                                                ")
    print("                                                                                ")
    print("                                                                                ")
    print("                                                                                ")
    print("================================================================================")
    print("================================================================================")
    print("                          13521026 - Kartini Copa                               ")
    print("================================================================================")
    print("================================================================================")

def print_brute_force(points):
    start = time.time()
    p1, p2, min_dist = brute_force.brute_force(points)
    end = time.time()
    print("================================================================================")
    print("                            Brute Force Algorithm                               ")
    print("================================================================================")
    print("Closest pair:")
    print("Point 1:", p1)
    print("Point 2:", p2)
    print("Closest distance:", min_dist)
    print("Total euclidean operation:", brute_force.eucl_brute)
    print("Execution time:", "{:.7f}".format(end - start))

def print_divide_conquer(points):
    start = time.time()
    p1, p2, min_dist = closest_pair.div_conquer(points)
    end = time.time()
    print("================================================================================")
    print("                         Divide and Conquer Algorithm                           ")
    print("================================================================================")
    print("Closest pair:")
    print("Point 1:", p1)
    print("Point 2:", p2)
    print("Closest distance:", min_dist)
    print("Total euclidean operation:", closest_pair.eucl_dnc)
    print("Execution time:", "{:.7f}".format(end - start))
```

Gambar 3.1.4 main.py

```python
if __name__ == '__main__':
    splash_screen()
    n = int(input("Enter the number of points: "))
    while n < 2:
        n = int(input("Enter the number of points: "))
    Rn = int(input("Enter the number of dimensions: "))
    points = generate_points(n, Rn)

    print_brute_force(points)
    print_divide_conquer(points)
    print("--------------------------------------------------------------------------------")
    print("Operation system:", platform.system(), platform.release())
    print("Processor:", platform.processor())
    print("Node:", platform.node())

    closest = closest_pair.div_conquer(points)
    closest_pair_points = [closest[0], closest[1]]

    x = [point[0] for point in points]
    y = [point[1] for point in points]
    if Rn == 2:
        colors = ['■#FF1493' if point in closest_pair_points else '■#33cccc' for point in points]
        plt.scatter(x, y, c=colors)
    elif Rn == 3:
        z = [point[2] for point in points]
        colors = ['■#FF1493' if point in closest_pair_points else '■#33cccc' for point in points]
        fig = plt.figure()
        ax = fig.add_subplot(projection='3d')
        ax.scatter(x, y, z, c=colors)
    else:
        print("Plotting can only be done in 2D or 3D space.")
    plt.show()
```

Gambar 3.1.5 main.py

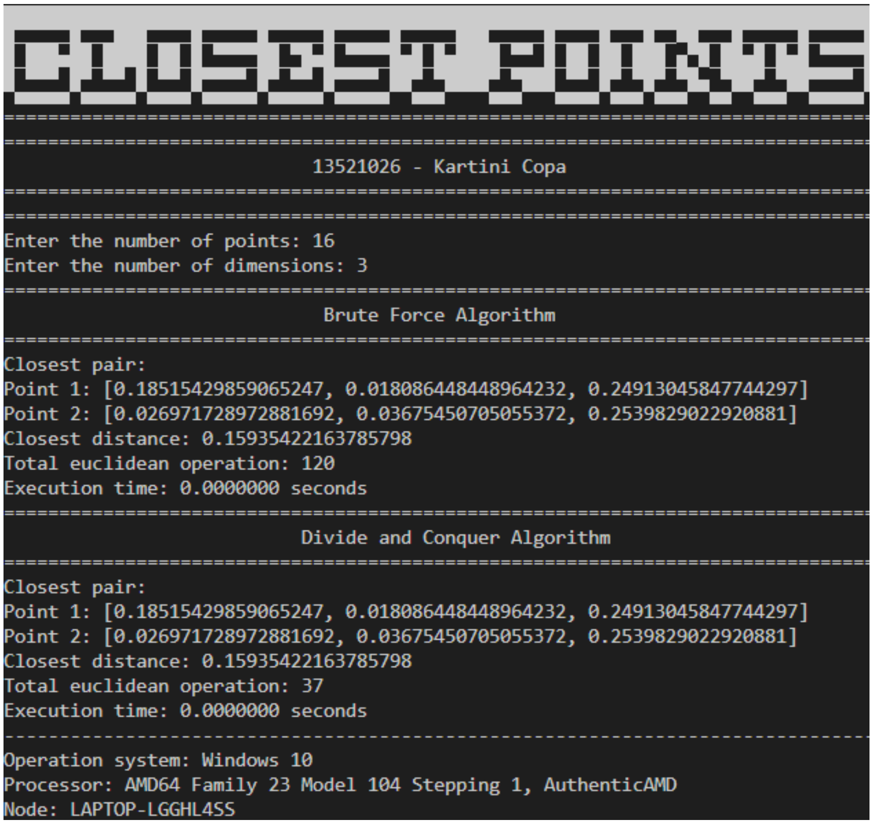
Bagian utama program dimulai dengan menampilkan splash screen dan meminta input dari pengguna berupa jumlah titik dan dimensi ruang. Setelah itu, himpunan titik dibangkitkan secara acak menggunakan fungsi generate_points(n, Rn). Setelah itu, program memanggil fungsi print_brute_force(points) dan print_divide_conquer(points) untuk mencetak hasil dari masing-masing algoritma. Program juga mencetak informasi mengenai sistem operasi, processor, dan node yang digunakan.

Terakhir, program akan membuat plot titik-titik yang dibangkitkan secara acak, di mana titik terdekat akan dicatat dengan warna berbeda. Plot hanya dapat dilakukan pada ruang 2D atau 3D, dan menggunakan warna merah muda (#FF1493) untuk titik terdekat dan warna biru laut (#33cccc) untuk titik-titik lainnya.

3.2 Pengujian

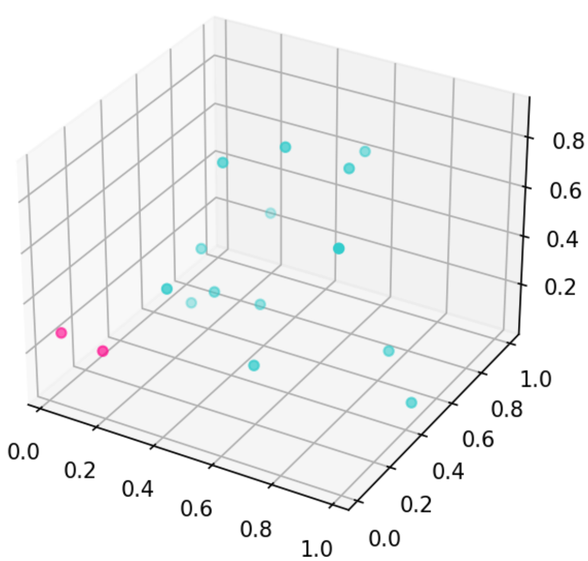
3.2.1 *Test Case* 1

Berikut merupakan pengujian terhadap 16 buah titik pada ruang dimensi 3 dan penggambarannya pada bidang 3D.



```
CLOSEST POINTS
================================================================================
================================================================================
                         13521026 - Kartini Copa
================================================================================
================================================================================
Enter the number of points: 16
Enter the number of dimensions: 3
================================================================================
                           Brute Force Algorithm
================================================================================
Closest pair:
Point 1: [0.18515429859065247, 0.018086448448964232, 0.24913045847744297]
Point 2: [0.026971728972881692, 0.03675450705055372, 0.2539829022920881]
Closest distance: 0.15935422163785798
Total euclidean operation: 120
Execution time: 0.0000000 seconds
================================================================================
                        Divide and Conquer Algorithm
================================================================================
Closest pair:
Point 1: [0.18515429859065247, 0.018086448448964232, 0.24913045847744297]
Point 2: [0.026971728972881692, 0.03675450705055372, 0.2539829022920881]
Closest distance: 0.15935422163785798
Total euclidean operation: 37
Execution time: 0.0000000 seconds
--------------------------------------------------------------------------------
Operation system: Windows 10
Processor: AMD64 Family 23 Model 104 Stepping 1, AuthenticAMD
Node: LAPTOP-LGGHL4SS
```

Gambar 3.2.1.1 *Test case* 1



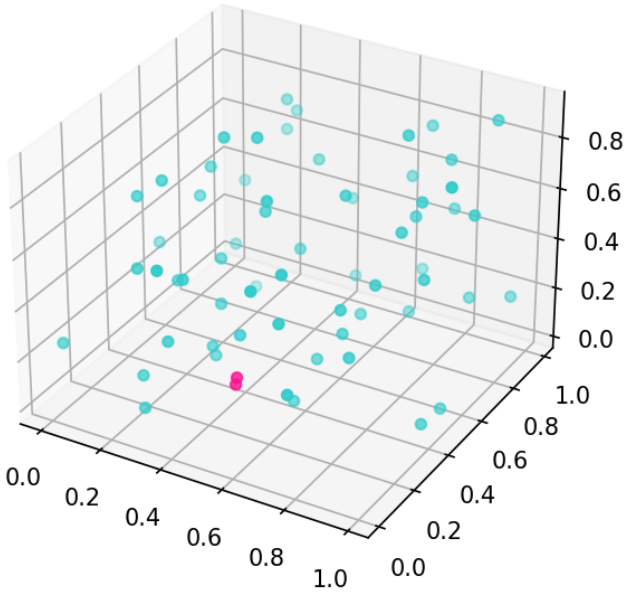
Gambar 3.2.1.2 3D *test case* 1

3.2.2 *Test Case* 2

Berikut merupakan pengujian terhadap 64 buah titik pada ruang dimensi 3 dan penggambarannya pada bidang 3D.

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗  ██████╗ ██╗███╗   ██╗████████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██╔═══██╗██║████╗  ██║╚══██╔══╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██████╔╝██║   ██║██║██╔██╗ ██║   ██║   ███████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██╔═══╝ ██║   ██║██║██║╚██╗██║   ██║   ╚════██║
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██║     ╚██████╔╝██║██║ ╚████║   ██║   ███████║
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═╝      ╚═════╝ ╚═╝╚═╝  ╚═══╝   ╚═╝   ╚══════╝
================================================================================
================================================================================
                            13521026 - Kartini Copa
================================================================================
================================================================================
Enter the number of points: 64
Enter the number of dimensions: 3
================================================================================
                              Brute Force Algorithm
================================================================================
Closest pair:
Point 1: [0.5361256421966075, 0.13073749530022494, 0.24189806868736452]
Point 2: [0.5384250760939244, 0.12115482179918602, 0.22168393872786218]
Closest distance: 0.022488354268271688
Total euclidean operation: 2016
Execution time: 0.0043881 seconds
================================================================================
                            Divide and Conquer Algorithm
================================================================================
Closest pair:
Point 1: [0.5361256421966075, 0.13073749530022494, 0.24189806868736452]
Point 2: [0.5384250760939244, 0.12115482179918602, 0.22168393872786218]
Closest distance: 0.022488354268271688
Total euclidean operation: 164
Execution time: 0.0000000 seconds
--------------------------------------------------------------------------------
Operation system: Windows 10
Processor: AMD64 Family 23 Model 104 Stepping 1, AuthenticAMD
Node: LAPTOP-LGGHL4SS
```

Gambar 3.2.2.1 *Test case* 2



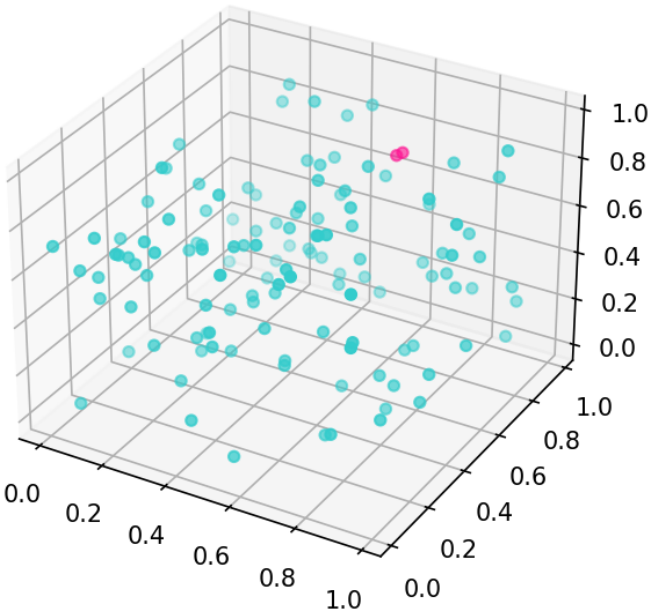
Gambar 3.2.3.2 3D *test case* 3

3.2.3 *Test Case* 3

Berikut merupakan pengujian terhadap 128 buah titik pada ruang dimensi 3 dan penggambarannya pada bidang 3D.

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗  ██████╗ ██╗███╗   ██╗████████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██╔═══██╗██║████╗  ██║╚══██╔══╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██████╔╝██║   ██║██║██╔██╗ ██║   ██║   ███████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██╔═══╝ ██║   ██║██║██║╚██╗██║   ██║   ╚════██║
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██║     ╚██████╔╝██║██║ ╚████║   ██║   ███████║
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═╝      ╚═════╝ ╚═╝╚═╝  ╚═══╝   ╚═╝   ╚══════╝
================================================================================
================================================================================
                            13521026 - Kartini Copa
================================================================================
================================================================================
Enter the number of points: 128
Enter the number of dimensions: 3
================================================================================
                            Brute Force Algorithm
================================================================================
Closest pair:
Point 1: [0.6206348231622235, 0.7916256030039318, 0.8154841877735589]
Point 2: [0.6388689354491873, 0.7925772849717132, 0.8330541399039604]
Closest distance: 0.025339529737664904
Total euclidean operation: 8128
Execution time: 0.0130513 seconds
================================================================================
                            Divide and Conquer Algorithm
================================================================================
Closest pair:
Point 1: [0.6206348231622235, 0.7916256030039318, 0.8154841877735589]
Point 2: [0.6388689354491873, 0.7925772849717132, 0.8330541399039604]
Closest distance: 0.025339529737664904
Total euclidean operation: 363
Execution time: 0.0009983 seconds
--------------------------------------------------------------------------------
Operation system: Windows 10
Processor: AMD64 Family 23 Model 104 Stepping 1, AuthenticAMD
Node: LAPTOP-LGGHL4SS
```

Gambar 3.2.3.1 *Test case* 3



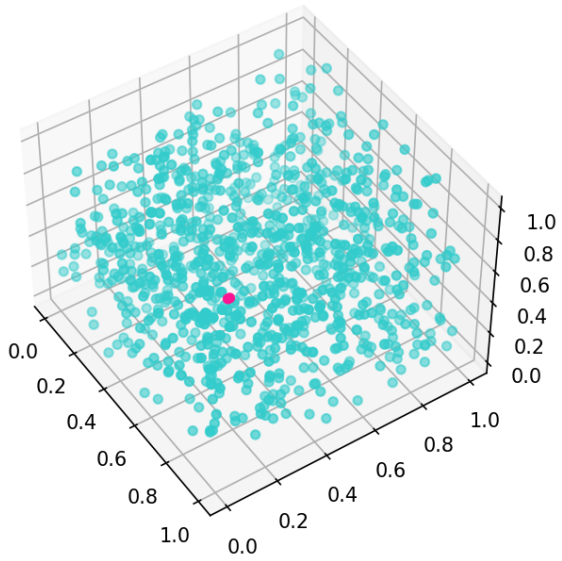
Gambar 3.2.3.2 3D *test case* 3

3.2.4 *Test Case* 4

Berikut merupakan pengujian terhadap 1000 buah titik pada ruang dimensi 3 dan penggambarannya pada bidang 3D.

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗  ██████╗ ██╗███╗   ██╗████████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██╔═══██╗██║████╗  ██║╚══██╔══╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██████╔╝██║   ██║██║██╔██╗ ██║   ██║   ███████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██╔═══╝ ██║   ██║██║██║╚██╗██║   ██║   ╚════██║
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██║     ╚██████╔╝██║██║ ╚████║   ██║   ███████║
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═╝      ╚═════╝ ╚═╝╚═╝  ╚═══╝   ╚═╝   ╚══════╝
================================================================================
================================================================================
                                13521026 - Kartini Copa
================================================================================
================================================================================
Enter the number of points: 1000
Enter the number of dimensions: 3
================================================================================
                              Brute Force Algorithm
================================================================================
Closest pair:
Point 1: [0.8451442324638507, 0.15743077241091175, 0.876379109255061]
Point 2: [0.8461288306762104, 0.1642211655100655, 0.8774077861374272]
Closest distance: 0.006938086783049865
Total euclidean operation: 499500
Execution time: 0.7466896 seconds
================================================================================
                           Divide and Conquer Algorithm
================================================================================
Closest pair:
Point 1: [0.8451442324638507, 0.15743077241091175, 0.876379109255061]
Point 2: [0.8461288306762104, 0.1642211655100655, 0.8774077861374272]
Closest distance: 0.006938086783049865
Total euclidean operation: 4423
Execution time: 0.0124161 seconds
--------------------------------------------------------------------------------
Operation system: Windows 10
Processor: AMD64 Family 23 Model 104 Stepping 1, AuthenticAMD
Node: LAPTOP-LGGHL4SS
```

Gambar 3.2.4.1 *Test case* 4



Gambar 3.2.4.2 3D *test case* 4

3.2.5 *Test Case* pada Ruang Dimensi Rn

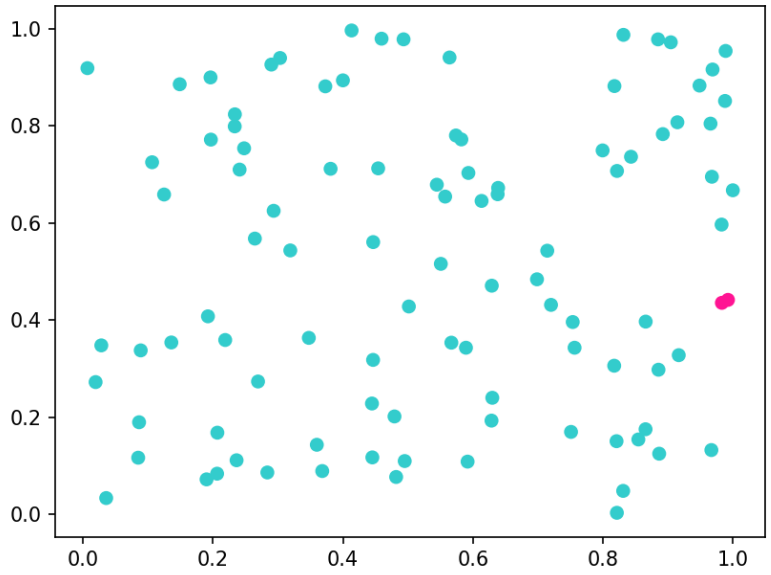
Berikut merupakan pengujian terhadap 16 buah titik pada ruang dimensi 7 tetapi untuk penggambarannya belum dapat divisualisasikan.

Gambar 3.2.5.1 *Test case* dimensi 7

Berikut merupakan pengujian terhadap 500 buah titik pada ruang dimensi 2 dan penggambarannya pada bidang 2D.

```
 ██████╗██╗      ██████╗ ███████╗███████╗███████╗████████╗    ██████╗  ██████╗ ██╗███╗   ██╗████████╗███████╗
██╔════╝██║     ██╔═══██╗██╔════╝██╔════╝██╔════╝╚══██╔══╝    ██╔══██╗██╔═══██╗██║████╗  ██║╚══██╔══╝██╔════╝
██║     ██║     ██║   ██║███████╗█████╗  ███████╗   ██║       ██████╔╝██║   ██║██║██╔██╗ ██║   ██║   ███████╗
██║     ██║     ██║   ██║╚════██║██╔══╝  ╚════██║   ██║       ██╔═══╝ ██║   ██║██║██║╚██╗██║   ██║   ╚════██║
╚██████╗███████╗╚██████╔╝███████║███████╗███████║   ██║       ██║     ╚██████╔╝██║██║ ╚████║   ██║   ███████║
 ╚═════╝╚══════╝ ╚═════╝ ╚══════╝╚══════╝╚══════╝   ╚═╝       ╚═╝      ╚═════╝ ╚═╝╚═╝  ╚═══╝   ╚═╝   ╚══════╝
================================================================================
================================================================================
                            13521026 - Kartini Copa
================================================================================
================================================================================
Enter the number of points: 100
Enter the number of dimensions: 2
================================================================================
                            Brute Force Algorithm
================================================================================
Closest pair:
Point 1: [0.9831420492144827, 0.4351904139543681]
Point 2: [0.9924866334789119, 0.44128508955104884]
Closest distance: 0.011156447718866142
Total euclidean operation: 4950
Execution time: 0.0070040 seconds
================================================================================
                            Divide and Conquer Algorithm
================================================================================
Closest pair:
Point 1: [0.9831420492144827, 0.4351904139543681]
Point 2: [0.9924866334789119, 0.44128508955104884]
Closest distance: 0.011156447718866142
Total euclidean operation: 103
Execution time: 0.0009816 seconds
--------------------------------------------------------------------------------
Operation system: Windows 10
Processor: AMD64 Family 23 Model 104 Stepping 1, AuthenticAMD
Node: LAPTOP-LGGHL4SS
```

Gambar 3.2.5.2 *Test case* dimensi 2



Gambar 3.2.5.3 2D *test case*

BAB IV

KESIMPULAN

5.1 Kesimpulan

- Algoritma *brute force* dapat diimplementasikan untuk mencari pasangan titik terdekat 3D tetapi waktu eksekusi yang lebih lama karena perlu memeriksa jarak Euclidean semua titik lain sehingga kompleksitas waktu algoritma adalah $O(n^2)$
- Algoritma *divide and conquer* dapat diimplementasikan untuk mencari pasangan titik terdekat 3D dengan waktu eksekusi yang lebih cepat dibandingkan dengan algoritma *brute force* karena membagi himpunan titik menjadi dua upa-himpunan dengan dengan titik tengah
- Kompleksitas algoritma *divide and conquer* dalam pencarian titik terdekat 3D adalah O(n log n)
- Penggambaran titik-titik hanya dapat dilakukan pada ruang dimensi 2 dan 3

5.2 Saran

- Pemahaman materi strategi algoritma *divide and conquer* yang lebih dalam dibutuhkan untuk pengerjaan program pencarian titik terdekat 3D karena sangat berkorelasi
- Pemahaman materi strategi algoritma *brute force* yang lebih dalam dibutuhkan untuk digunakan sebagai pembanding

5.3 Refleksi

- Melalui tugas kecil ini penulis menyadari secara nyata implementasi dari materi Strategi Algoritma IF2211
- Melalui tugas kecil ini penulis menyadari secara nyata perbedaan algoritma *brute force* dan algoritma *divide and conquer*

DAFTAR REFERENSI

[1] R. Munir (2022). Algoritma Brute Force Bagian 1 [Powerpoint Slides]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf

LAMPIRAN

Link repository: https://github.com/kartinicopa/Tucil2_13521026.git

Checklist:

No.	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa ada kesalahan	✔	
2	Program berhasil *running*	✔	
3	Program dapat menerima masukan dan dan menuliskan luaran	✔	
4	Luaran program sudah benar (solusi *closest pair* benar)	✔	
5	Bonus 1 dikerjakan	✔	
6	Bonus 2 dikerjakan	✔	