TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA 2022/2023

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

Disusun oleh:

Rayhan Hanif Maulana Pradana/13521112

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022/2023

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

Pencarian satu pasang titik terdekat merupakan salah satu masalah klasik yang penerapannya banyak di kehidupan nyata, seperti *Air Traffic Control* yang mengatur jalur udara terutama di sekitar bandar udara. Misalkan terdapat *k* buah titik pada ruang *N*-dimensi. Setiap titik *P* di dalam ruang dinyatakan dengan koordinat *P* = (u_1, u_2, u_3, …, u_n). Akan dicari jarak *d* dari dua titik terdekat P_1 = (u_{11}, u_{21}, u_{31}, …, u_{n1}) dan P_2 = (u_{12}, u_{22}, u_{32}, …, u_{n2}). Jarak *d* dihitung dengan rumus Euclidean berikut:

$$d = \sqrt{(u_{11} - u_{12})^2 + (u_{21} - u_{22})^2 + (u_{31} - u_{32})^2 + \cdots + (u_{n1} - u_{n2})^2}$$

Akan dibuat sebuah program yang menerima masukan berupa:

- Dimensi *n* (*n* > 0)
- *k* buah titik (*k* > 1)

Dengan luaran program berupa:

- Semua titik sejumlah *k* yang dibangkitkan secara acak (terurut berdasarkan sumbu-x atau u_{11})
- Sepasang titik terdekat dengan jaraknya menggunakan algoritma *Brute Force*
- Sepasang titik terdekat dengan jaraknya menggunakan algoritma *Divide and Conquer*
- Banyaknya perhitungan operasi rumus Euclidean setiap algoritma
- Waktu eksekusi setiap algoritma, dengan spesifikasi mesin yang digunakan
- Penggambaran/visualisasi untuk 1D, 2D, dan 3D

BAB II

ALGORITMA *DIVIDE AND CONQUER*

UNTUK PENCARIAN SEPASANG TITIK TERDEKAT *N*-DIMENSI

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

1. Masukan berupa larik bertipe titik di ruang *N*-Dimensi yang telah terurut berdasarkan nilai sumbu-x atau u_{11}.
2. Kasus basisnya adalah ketika hanya terdapat dua titik dimana jaraknya dapat langsung dihitung dengan rumus Euclidean, atau tiga titik yang jarak terdekatnya juga dapat langsung dihitung dalam tiga perhitungan Euclidean.
3. Jika belum mencapai kasus basis, rekursi dilakukan dengan membagi larik menjadi dua bagian/daerah sama rata secara terus-menerus.
4. Masing-masing bagian/daerah akan dicari untuk jarak terkecil dari titik-titiknya, dan diambil nilai minimum dari kedua bagian daerah.
5. Akan diambil nilai delta, yaitu nilai jarak minimum dari kedua bagian daerah. Ditarik sebuah daerah dari pembagi sejauh delta. Titik yang berada di daerah tersebut (daerah dengan jarak delta dari pembagi) dicari jarak terkecilnya.
6. Dilakukan berulang kali secara rekursif hingga mendapat jarak terkecil dari himpunan titik tersebut.

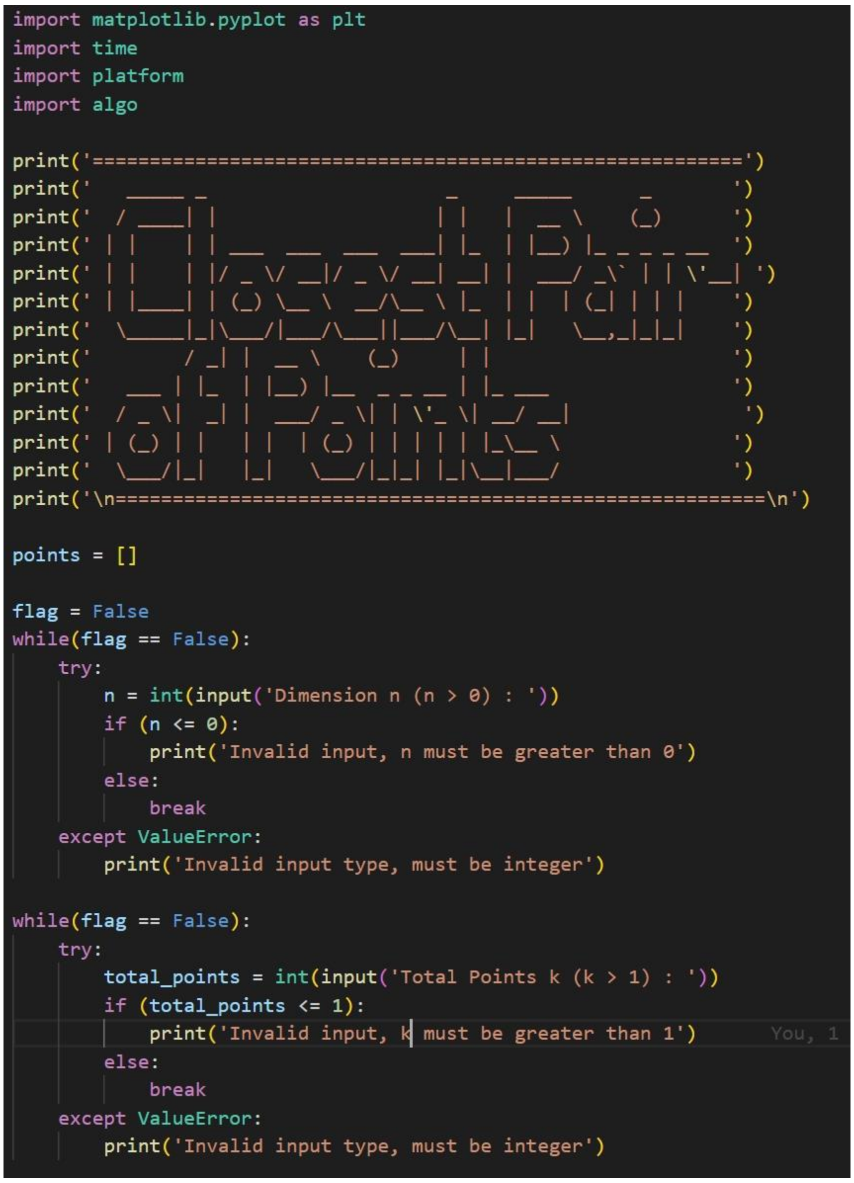
BAB III

SOURCE CODE

Implementasi dilakukan dengan menggunakan bahasa pemrograman Python3.

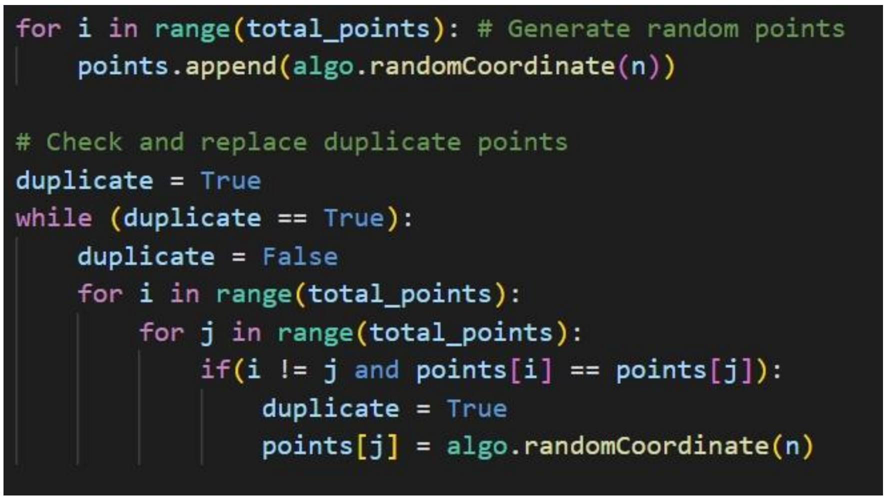
3.1 main.py

3.1.1 Tampilan utama



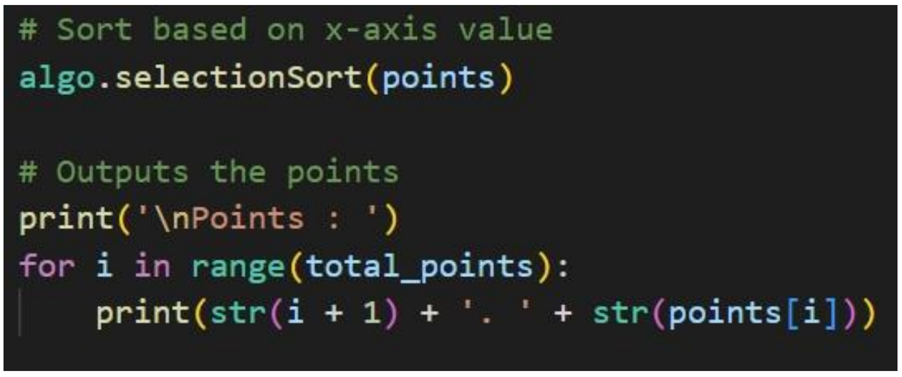
```python
import matplotlib.pyplot as plt
import time
import platform
import algo

print('=========================================================')
print('   _____ _                     _     _____      _        ')
print('  / ____| |                   | |   |  __ \    (_)       ')
print(' | |    | | ___  ___  ___  ___| |_  | |__) |_ _ _ _ __   ')
print(' | |    | |/ _ \/ __|/ _ \/ __| __| |  ___/ _` | | \'__| ')
print(' | |____| | (_) \__ \  __/\__ \ |_  | |  | (_| | | |     ')
print('  \_____|_|\___/|___/\___||___/\__| |_|   \__,_|_|_|     ')
print('        / _| |  __ \    (_)     | |                      ')
print('   ___ | |_  | |__) |__  _ _ __ | |_ ___                 ')
print('  / _ \|  _| |  ___/ _ \| | \'_ \| __/ __|                ')
print(' | (_) | |   | |  | (_) | | | | | |_\__ \                ')
print('  \___/|_|   |_|   \___/|_|_| |_|\__|___/                ')
print('\n=========================================================\n')

points = []

flag = False
while(flag == False):
    try:
        n = int(input('Dimension n (n > 0) : '))
        if (n <= 0):
            print('Invalid input, n must be greater than 0')
        else:
            break
    except ValueError:
        print('Invalid input type, must be integer')

while(flag == False):
    try:
        total_points = int(input('Total Points k (k > 1) : '))
        if (total_points <= 1):
            print('Invalid input, k must be greater than 1')        You, 1
        else:
            break
    except ValueError:
        print('Invalid input type, must be integer')
```

3.1.2 Membangkitkan titik secara acak



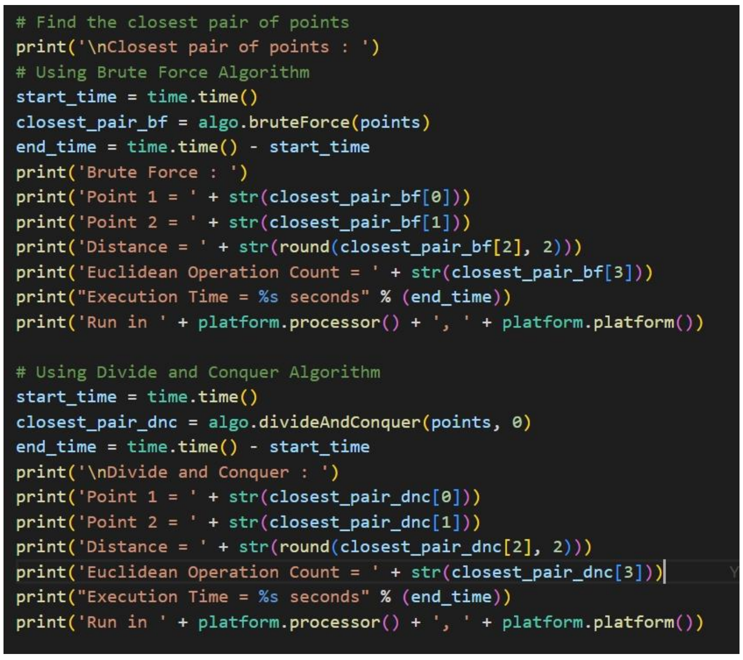
```python
for i in range(total_points): # Generate random points
    points.append(algo.randomCoordinate(n))

# Check and replace duplicate points
duplicate = True
while (duplicate == True):
    duplicate = False
    for i in range(total_points):
        for j in range(total_points):
            if(i != j and points[i] == points[j]):
                duplicate = True
                points[j] = algo.randomCoordinate(n)
```

3.1.3 Sort dan keluaran titik-titik yang telah dibangkitkan



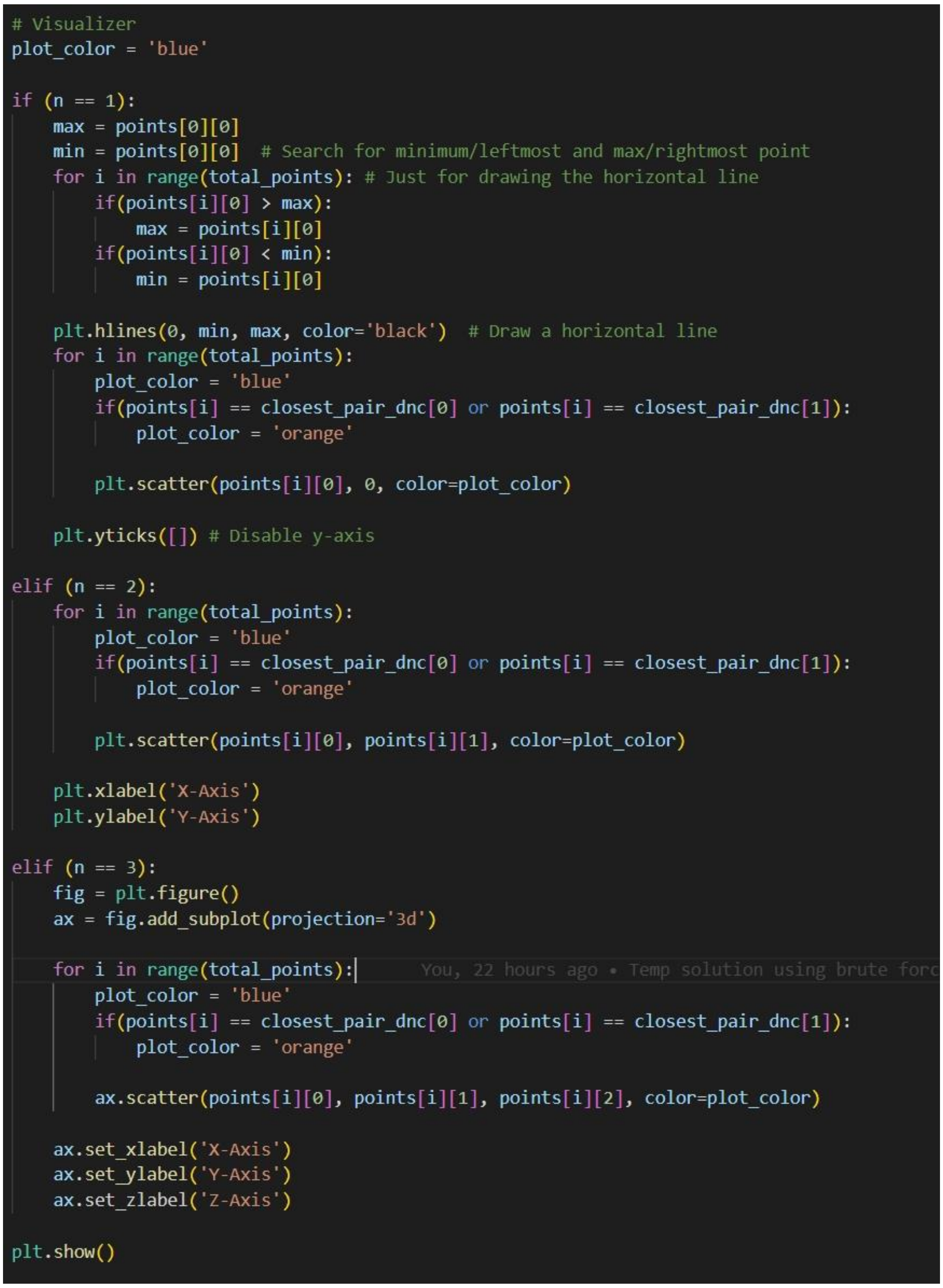
```python
# Sort based on x-axis value
algo.selectionSort(points)

# Outputs the points
print('\nPoints : ')
for i in range(total_points):
    print(str(i + 1) + '. ' + str(points[i]))
```

3.1.4 Perbandingan Algoritma *Brute Force* dengan Algoritma *Divide and Conquer*



```python
# Find the closest pair of points
print('\nClosest pair of points : ')
# Using Brute Force Algorithm
start_time = time.time()
closest_pair_bf = algo.bruteForce(points)
end_time = time.time() - start_time
print('Brute Force : ')
print('Point 1 = ' + str(closest_pair_bf[0]))
print('Point 2 = ' + str(closest_pair_bf[1]))
print('Distance = ' + str(round(closest_pair_bf[2], 2)))
print('Euclidean Operation Count = ' + str(closest_pair_bf[3]))
print("Execution Time = %s seconds" % (end_time))
print('Run in ' + platform.processor() + ', ' + platform.platform())

# Using Divide and Conquer Algorithm
start_time = time.time()
closest_pair_dnc = algo.divideAndConquer(points, 0)
end_time = time.time() - start_time
print('\nDivide and Conquer : ')
print('Point 1 = ' + str(closest_pair_dnc[0]))
print('Point 2 = ' + str(closest_pair_dnc[1]))
print('Distance = ' + str(round(closest_pair_dnc[2], 2)))
print('Euclidean Operation Count = ' + str(closest_pair_dnc[3]))
print("Execution Time = %s seconds" % (end_time))
print('Run in ' + platform.processor() + ', ' + platform.platform())
```

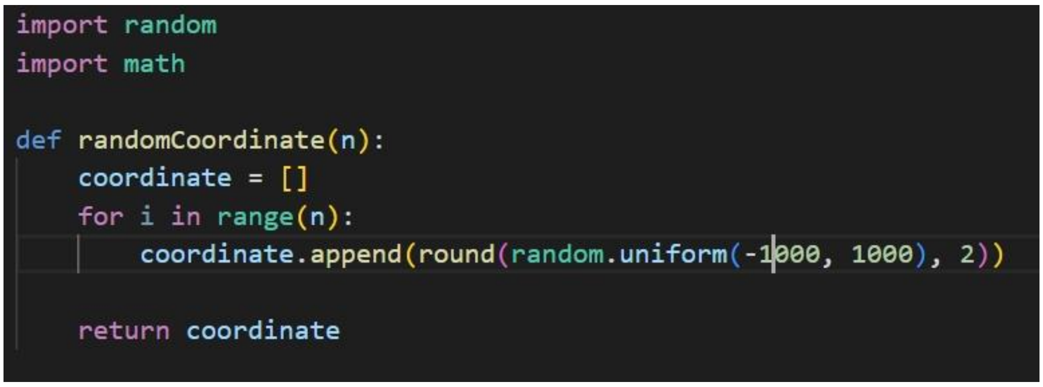
3.1.5 Visualisasi



```python
# Visualizer
plot_color = 'blue'

if (n == 1):
    max = points[0][0]
    min = points[0][0]  # Search for minimum/leftmost and max/rightmost point
    for i in range(total_points): # Just for drawing the horizontal line
        if(points[i][0] > max):
            max = points[i][0]
        if(points[i][0] < min):
            min = points[i][0]

    plt.hlines(0, min, max, color='black')  # Draw a horizontal line
    for i in range(total_points):
        plot_color = 'blue'
        if(points[i] == closest_pair_dnc[0] or points[i] == closest_pair_dnc[1]):
            plot_color = 'orange'

        plt.scatter(points[i][0], 0, color=plot_color)

    plt.yticks([]) # Disable y-axis

elif (n == 2):
    for i in range(total_points):
        plot_color = 'blue'
        if(points[i] == closest_pair_dnc[0] or points[i] == closest_pair_dnc[1]):
            plot_color = 'orange'

        plt.scatter(points[i][0], points[i][1], color=plot_color)

    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')

elif (n == 3):
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    for i in range(total_points):        You, 22 hours ago • Temp solution using brute forc
        plot_color = 'blue'
        if(points[i] == closest_pair_dnc[0] or points[i] == closest_pair_dnc[1]):
            plot_color = 'orange'

        ax.scatter(points[i][0], points[i][1], points[i][2], color=plot_color)

    ax.set_xlabel('X-Axis')
    ax.set_ylabel('Y-Axis')
    ax.set_zlabel('Z-Axis')

plt.show()
```

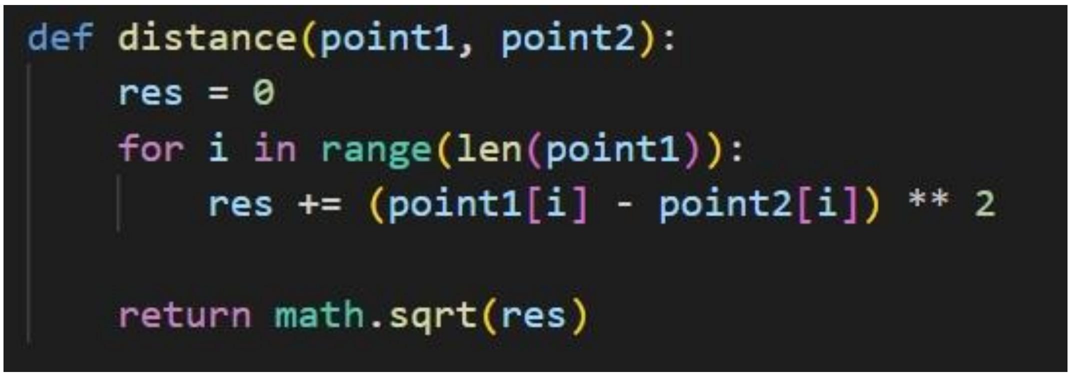
3.2 algo.py

3.2.1 Pembangkit koordinat acak



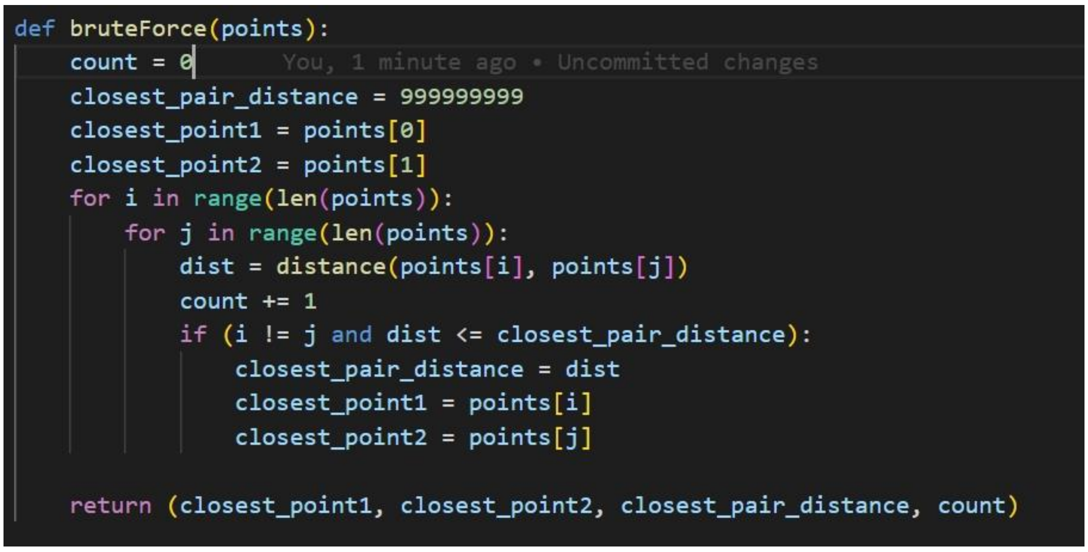
```python
import random
import math

def randomCoordinate(n):
    coordinate = []
    for i in range(n):
        coordinate.append(round(random.uniform(-1000, 1000), 2))

    return coordinate
```

3.2.2 Euclidean distance



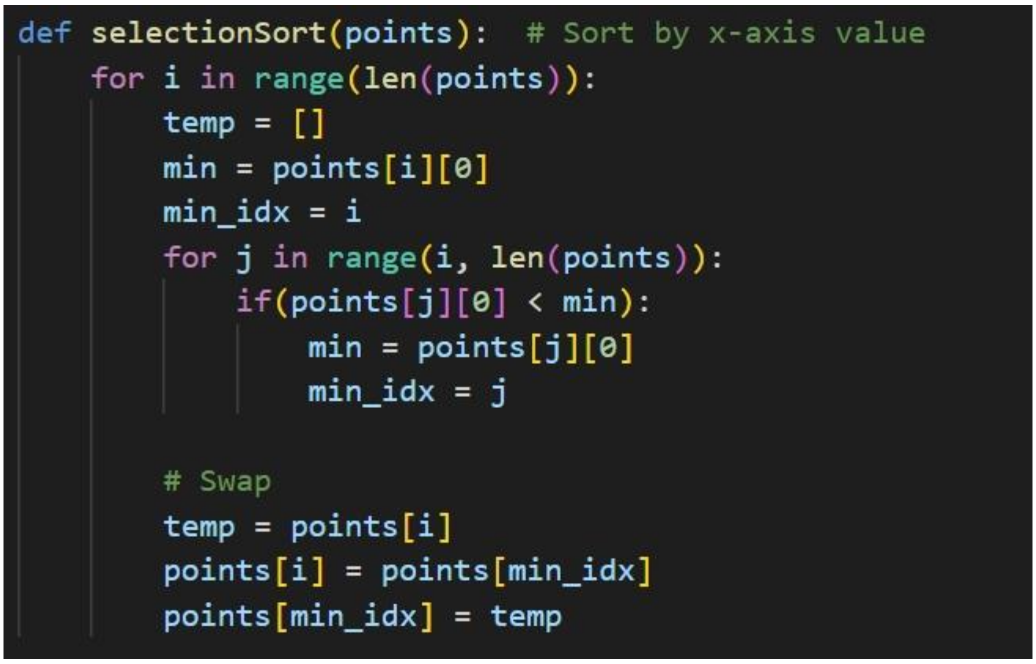
```python
def distance(point1, point2):
    res = 0
    for i in range(len(point1)):
        res += (point1[i] - point2[i]) ** 2

    return math.sqrt(res)
```

3.2.3 *Brute Force*



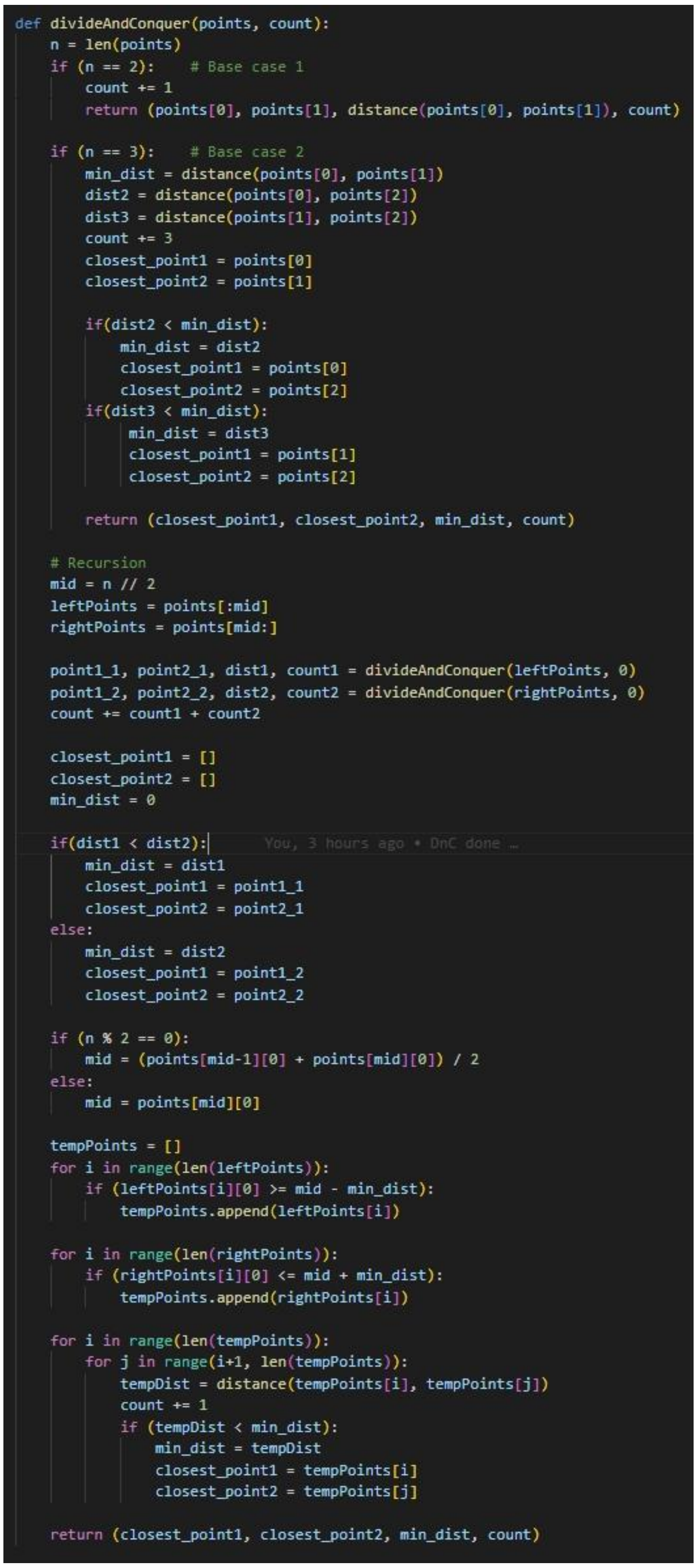
```python
def bruteForce(points):
    count = 0        You, 1 minute ago • Uncommitted changes
    closest_pair_distance = 999999999
    closest_point1 = points[0]
    closest_point2 = points[1]
    for i in range(len(points)):
        for j in range(len(points)):
            dist = distance(points[i], points[j])
            count += 1
            if (i != j and dist <= closest_pair_distance):
                closest_pair_distance = dist
                closest_point1 = points[i]
                closest_point2 = points[j]

    return (closest_point1, closest_point2, closest_pair_distance, count)
```

3.2.4 *Selection Sort*



```python
def selectionSort(points):  # Sort by x-axis value
    for i in range(len(points)):
        temp = []
        min = points[i][0]
        min_idx = i
        for j in range(i, len(points)):
            if(points[j][0] < min):
                min = points[j][0]
                min_idx = j

        # Swap
        temp = points[i]
        points[i] = points[min_idx]
        points[min_idx] = temp
```

3.2.5 *Divide and Conquer*



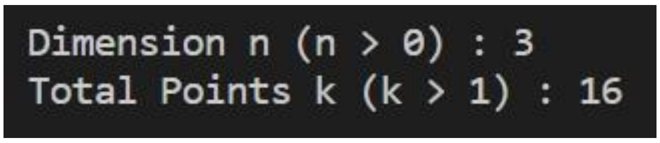
```python
def divideAndConquer(points, count):
    n = len(points)
    if (n == 2):    # Base case 1
        count += 1
        return (points[0], points[1], distance(points[0], points[1]), count)

    if (n == 3):    # Base case 2
        min_dist = distance(points[0], points[1])
        dist2 = distance(points[0], points[2])
        dist3 = distance(points[1], points[2])
        count += 3
        closest_point1 = points[0]
        closest_point2 = points[1]

        if(dist2 < min_dist):
            min_dist = dist2
            closest_point1 = points[0]
            closest_point2 = points[2]
        if(dist3 < min_dist):
             min_dist = dist3
             closest_point1 = points[1]
             closest_point2 = points[2]

        return (closest_point1, closest_point2, min_dist, count)

    # Recursion
    mid = n // 2
    leftPoints = points[:mid]
    rightPoints = points[mid:]

    point1_1, point2_1, dist1, count1 = divideAndConquer(leftPoints, 0)
    point1_2, point2_2, dist2, count2 = divideAndConquer(rightPoints, 0)
    count += count1 + count2

    closest_point1 = []
    closest_point2 = []
    min_dist = 0

    if(dist1 < dist2):        You, 3 hours ago • DnC done ...
        min_dist = dist1
        closest_point1 = point1_1
        closest_point2 = point2_1
    else:
        min_dist = dist2
        closest_point1 = point1_2
        closest_point2 = point2_2

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

    tempPoints = []
    for i in range(len(leftPoints)):
        if (leftPoints[i][0] >= mid - min_dist):
            tempPoints.append(leftPoints[i])

    for i in range(len(rightPoints)):
        if (rightPoints[i][0] <= mid + min_dist):
            tempPoints.append(rightPoints[i])

    for i in range(len(tempPoints)):
        for j in range(i+1, len(tempPoints)):
            tempDist = distance(tempPoints[i], tempPoints[j])
            count += 1
            if (tempDist < min_dist):
                min_dist = tempDist
                closest_point1 = tempPoints[i]
                closest_point2 = tempPoints[j]

    return (closest_point1, closest_point2, min_dist, count)
```

BAB IV

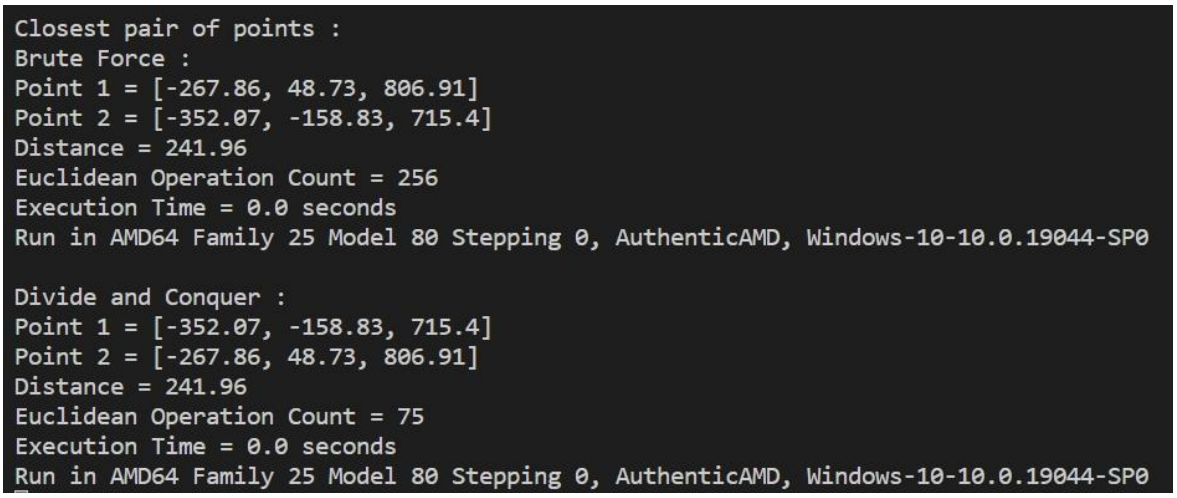
INPUT DAN OUTPUT PROGRAM

4.1 Jumlah Titik = 16

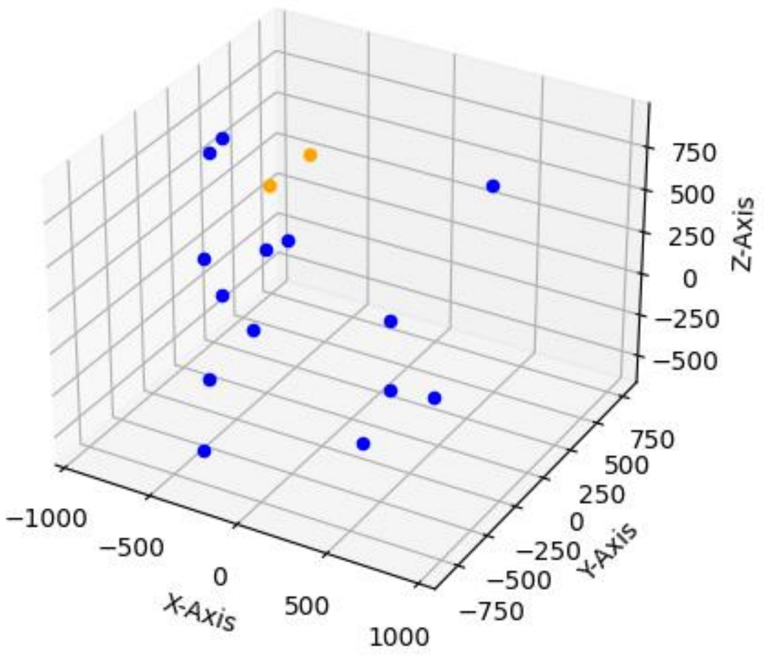
4.1.1 3D



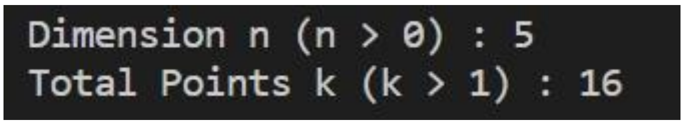
```
Dimension n (n > 0) : 3
Total Points k (k > 1) : 16
```



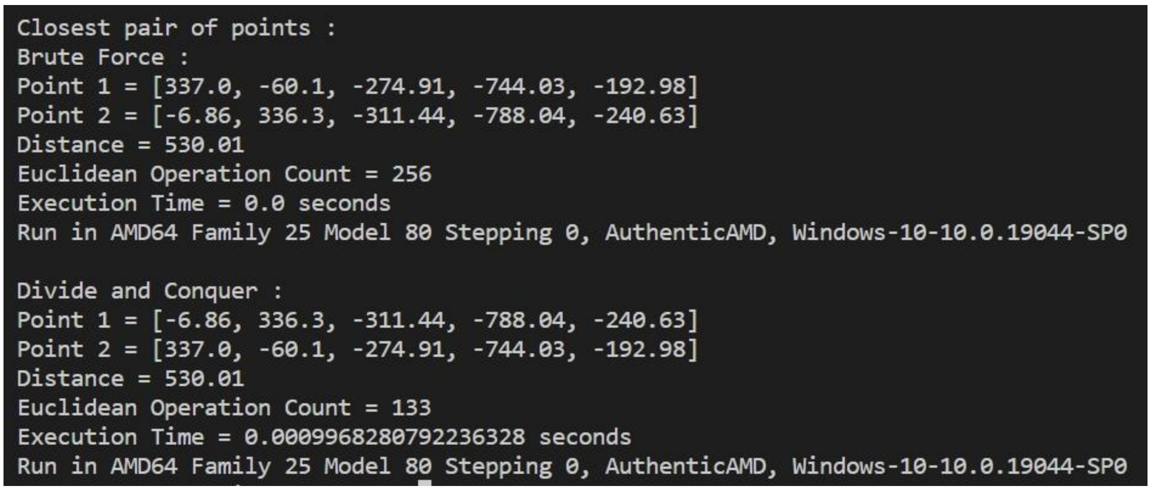
```
Closest pair of points :
Brute Force :
Point 1 = [-267.86, 48.73, 806.91]
Point 2 = [-352.07, -158.83, 715.4]
Distance = 241.96
Euclidean Operation Count = 256
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-352.07, -158.83, 715.4]
Point 2 = [-267.86, 48.73, 806.91]
Distance = 241.96
Euclidean Operation Count = 75
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```



4.1.2 5D

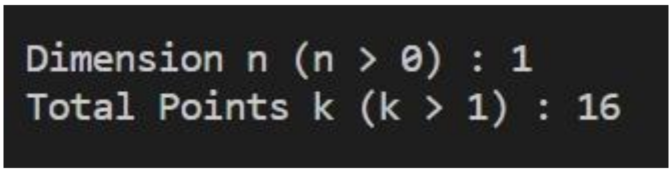


```
Dimension n (n > 0) : 5
Total Points k (k > 1) : 16
```

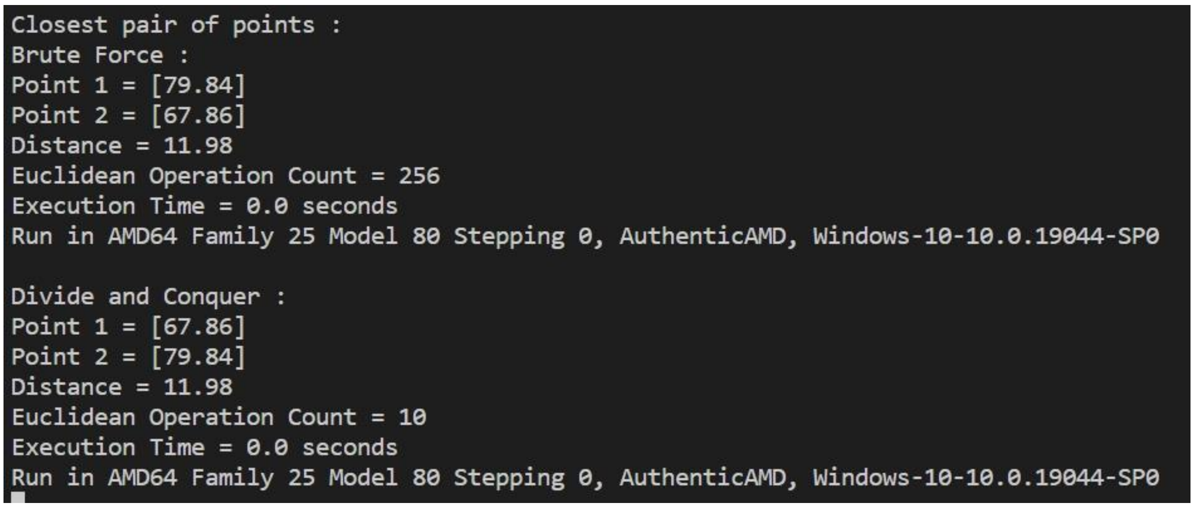


```
Closest pair of points :
Brute Force :
Point 1 = [337.0, -60.1, -274.91, -744.03, -192.98]
Point 2 = [-6.86, 336.3, -311.44, -788.04, -240.63]
Distance = 530.01
Euclidean Operation Count = 256
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-6.86, 336.3, -311.44, -788.04, -240.63]
Point 2 = [337.0, -60.1, -274.91, -744.03, -192.98]
Distance = 530.01
Euclidean Operation Count = 133
Execution Time = 0.0009968280792236328 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

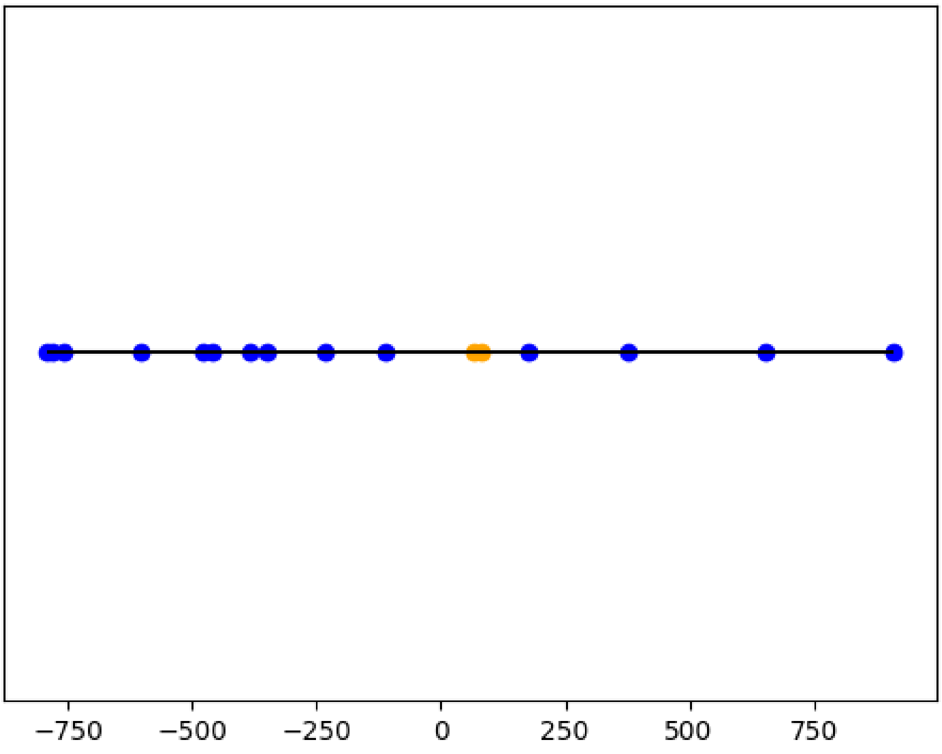
4.1.3 1D



```
Dimension n (n > 0) : 1
Total Points k (k > 1) : 16
```

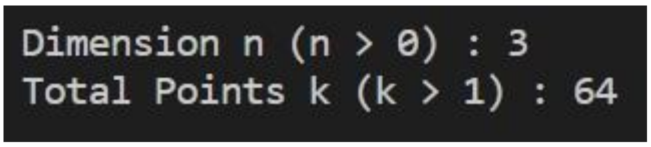


```
Closest pair of points :
Brute Force :
Point 1 = [79.84]
Point 2 = [67.86]
Distance = 11.98
Euclidean Operation Count = 256
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [67.86]
Point 2 = [79.84]
Distance = 11.98
Euclidean Operation Count = 10
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

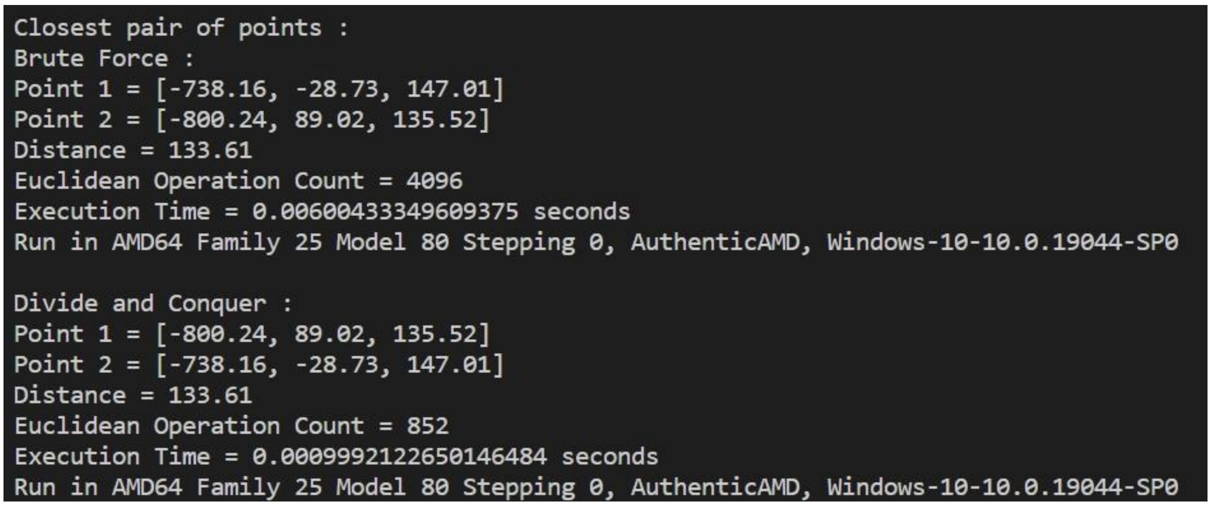


4.2 Jumlah Titik = 64

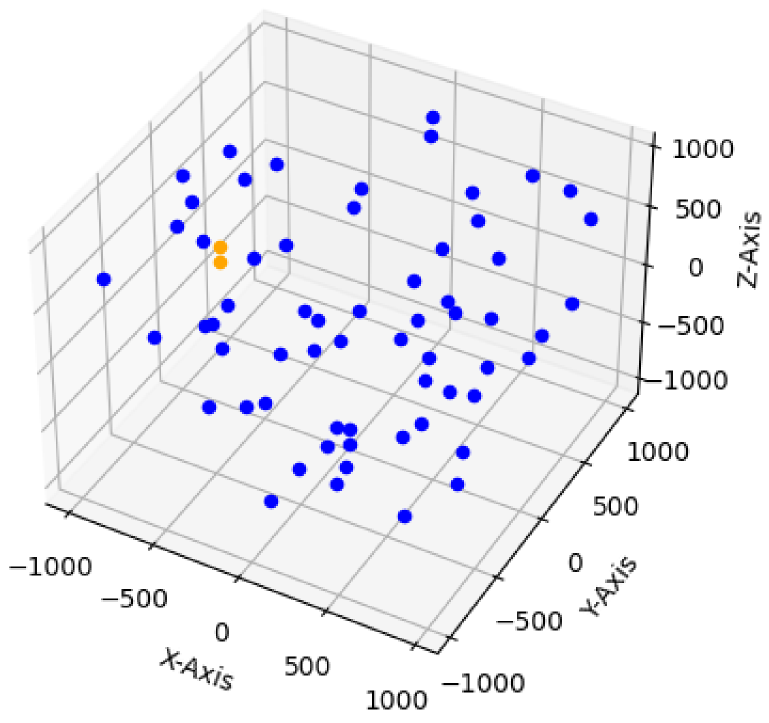
4.2.1 3D



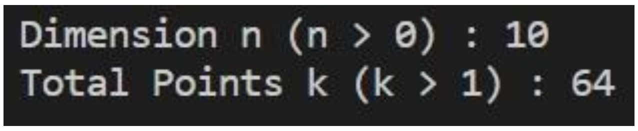
```
Dimension n (n > 0) : 3
Total Points k (k > 1) : 64
```



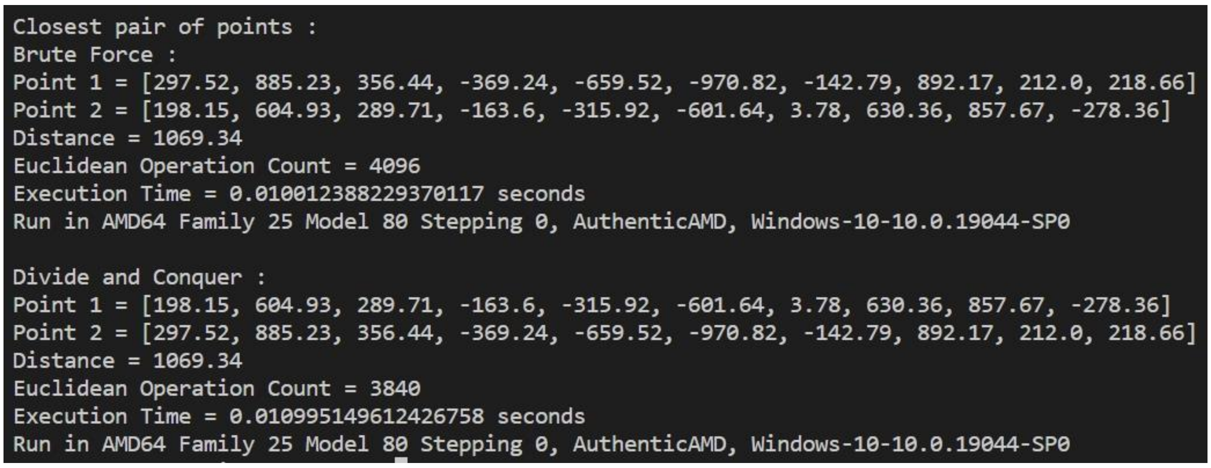
```
Closest pair of points :
Brute Force :
Point 1 = [-738.16, -28.73, 147.01]
Point 2 = [-800.24, 89.02, 135.52]
Distance = 133.61
Euclidean Operation Count = 4096
Execution Time = 0.00600433349609375 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-800.24, 89.02, 135.52]
Point 2 = [-738.16, -28.73, 147.01]
Distance = 133.61
Euclidean Operation Count = 852
Execution Time = 0.0009992122650146484 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```



4.2.2 10D

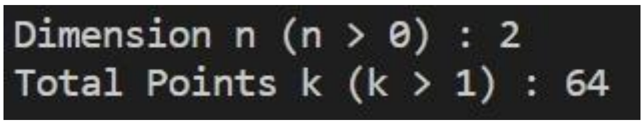


```
Dimension n (n > 0) : 10
Total Points k (k > 1) : 64
```

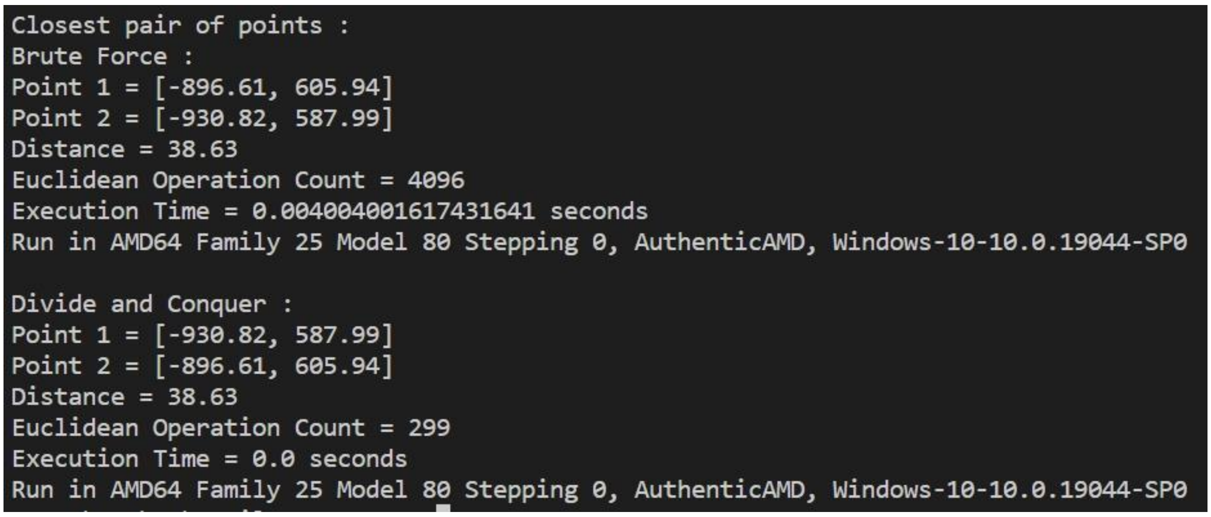


```
Closest pair of points :
Brute Force :
Point 1 = [297.52, 885.23, 356.44, -369.24, -659.52, -970.82, -142.79, 892.17, 212.0, 218.66]
Point 2 = [198.15, 604.93, 289.71, -163.6, -315.92, -601.64, 3.78, 630.36, 857.67, -278.36]
Distance = 1069.34
Euclidean Operation Count = 4096
Execution Time = 0.010012388229370117 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [198.15, 604.93, 289.71, -163.6, -315.92, -601.64, 3.78, 630.36, 857.67, -278.36]
Point 2 = [297.52, 885.23, 356.44, -369.24, -659.52, -970.82, -142.79, 892.17, 212.0, 218.66]
Distance = 1069.34
Euclidean Operation Count = 3840
Execution Time = 0.010995149612426758 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

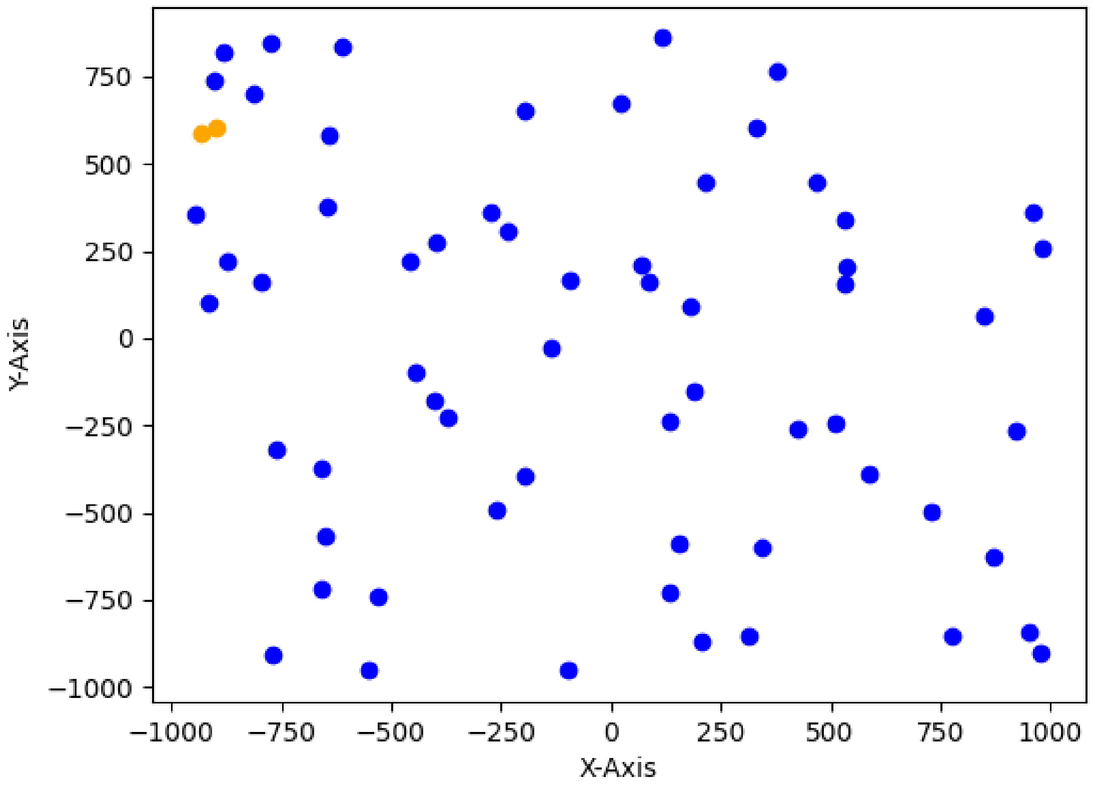
4.2.3 2D



```
Dimension n (n > 0) : 2
Total Points k (k > 1) : 64
```

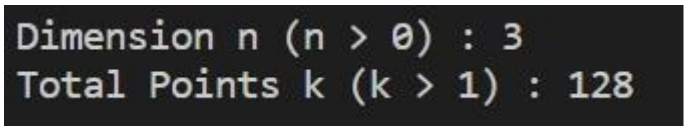


```
Closest pair of points :
Brute Force :
Point 1 = [-896.61, 605.94]
Point 2 = [-930.82, 587.99]
Distance = 38.63
Euclidean Operation Count = 4096
Execution Time = 0.004004001617431641 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-930.82, 587.99]
Point 2 = [-896.61, 605.94]
Distance = 38.63
Euclidean Operation Count = 299
Execution Time = 0.0 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

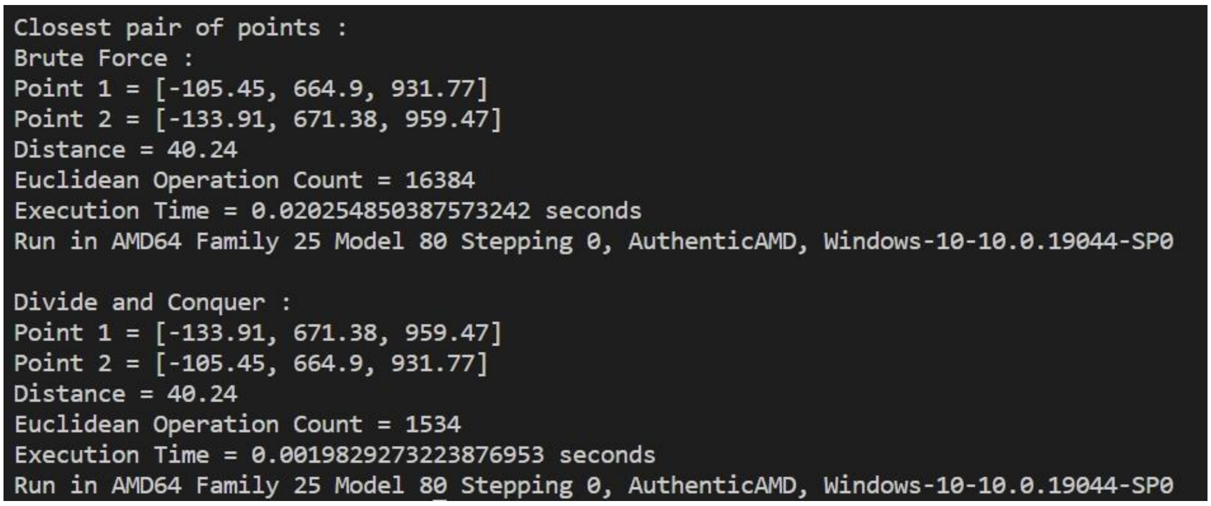


4.3 Jumlah Titik = 128

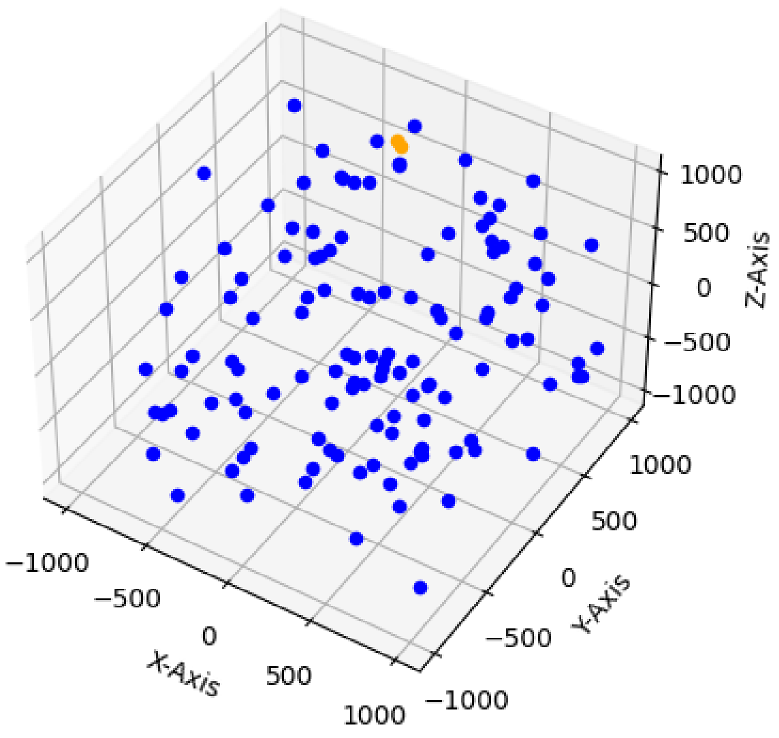
4.3.1 3D



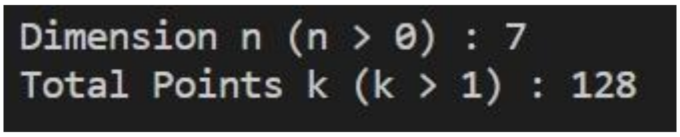
```
Dimension n (n > 0) : 3
Total Points k (k > 1) : 128
```



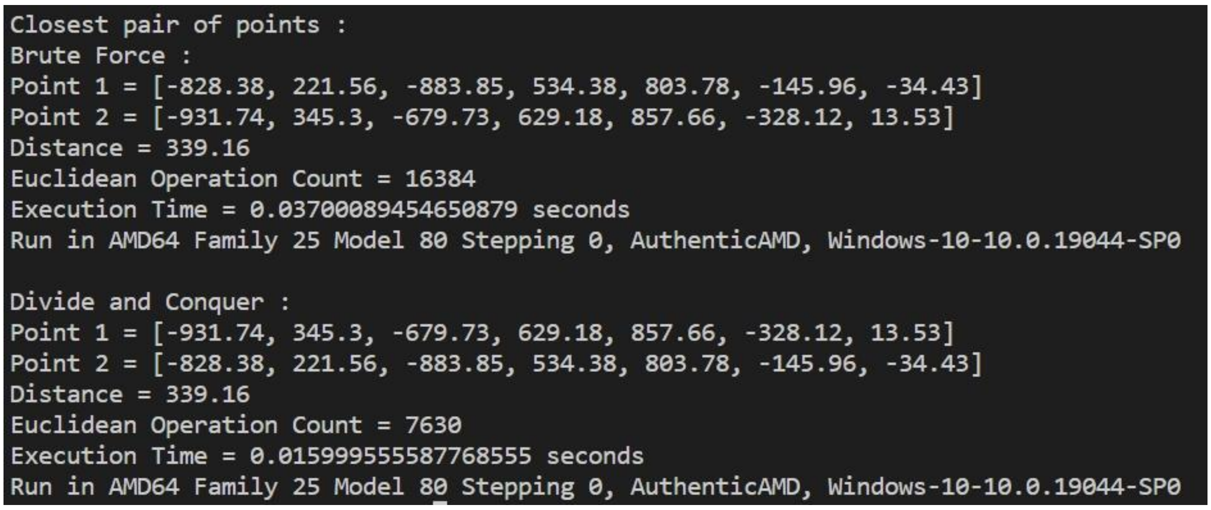
```
Closest pair of points :
Brute Force :
Point 1 = [-105.45, 664.9, 931.77]
Point 2 = [-133.91, 671.38, 959.47]
Distance = 40.24
Euclidean Operation Count = 16384
Execution Time = 0.020254850387573242 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-133.91, 671.38, 959.47]
Point 2 = [-105.45, 664.9, 931.77]
Distance = 40.24
Euclidean Operation Count = 1534
Execution Time = 0.0019829273223876953 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```



4.3.2 7D



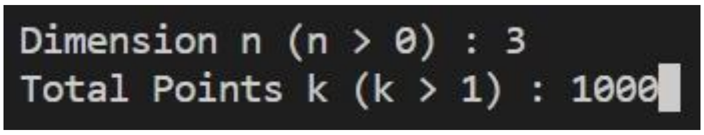
```
Dimension n (n > 0) : 7
Total Points k (k > 1) : 128
```



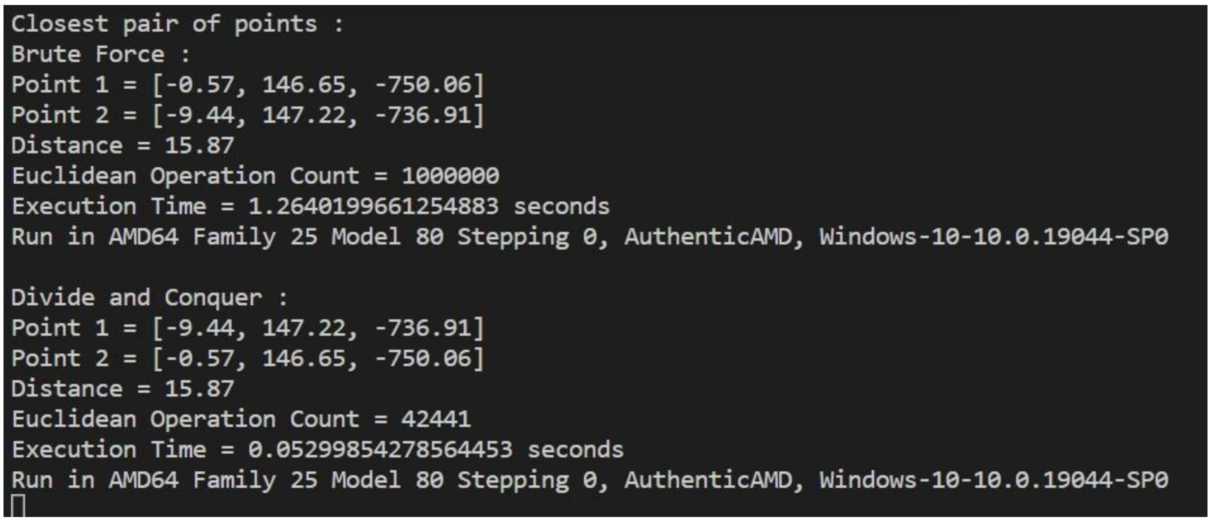
```
Closest pair of points :
Brute Force :
Point 1 = [-828.38, 221.56, -883.85, 534.38, 803.78, -145.96, -34.43]
Point 2 = [-931.74, 345.3, -679.73, 629.18, 857.66, -328.12, 13.53]
Distance = 339.16
Euclidean Operation Count = 16384
Execution Time = 0.03700089454650879 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-931.74, 345.3, -679.73, 629.18, 857.66, -328.12, 13.53]
Point 2 = [-828.38, 221.56, -883.85, 534.38, 803.78, -145.96, -34.43]
Distance = 339.16
Euclidean Operation Count = 7630
Execution Time = 0.015999555587768555 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

4.4 Jumlah Titik = 1000

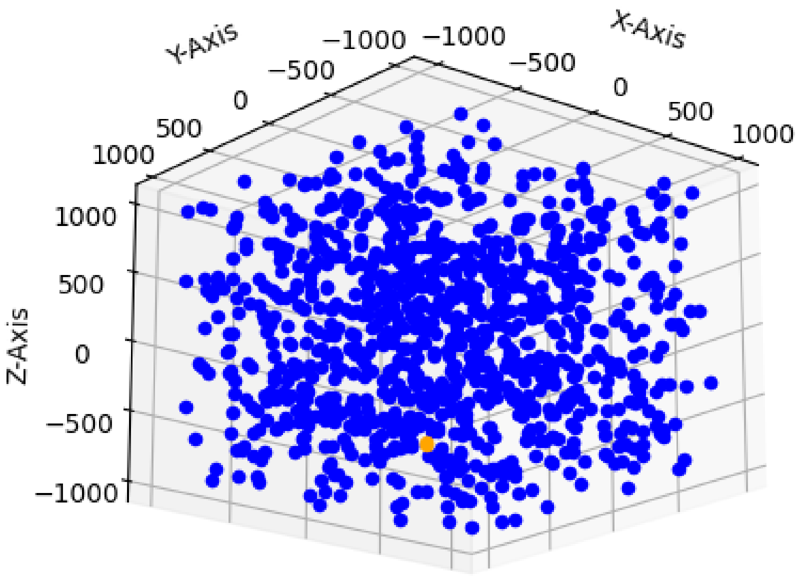
4.4.1 3D



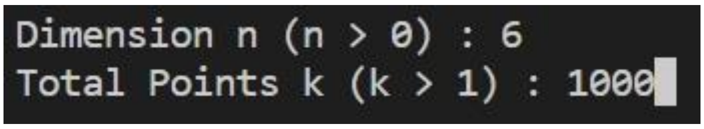
```
Dimension n (n > 0) : 3
Total Points k (k > 1) : 1000
```



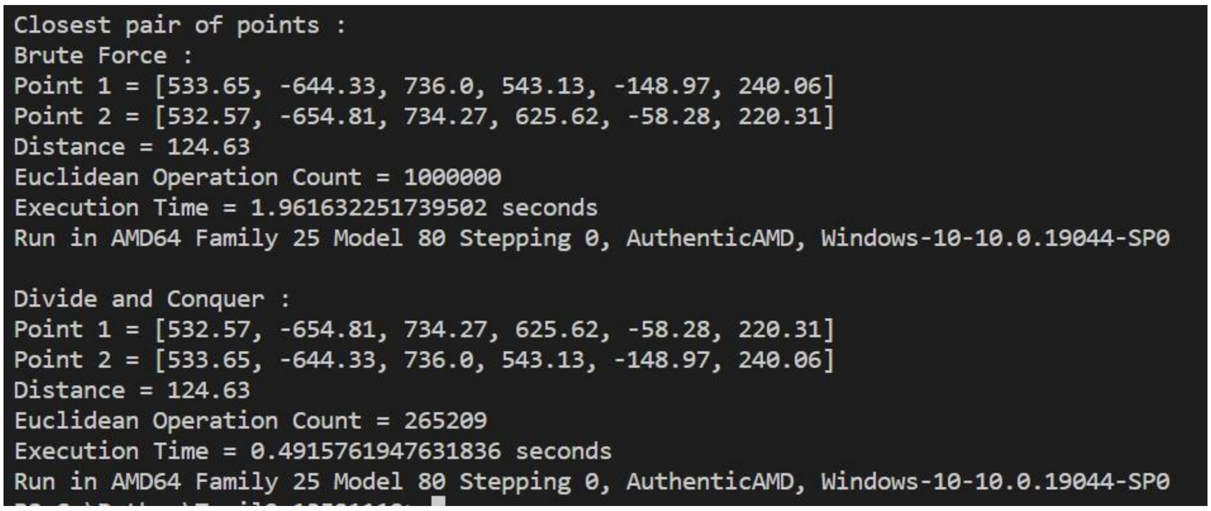
```
Closest pair of points :
Brute Force :
Point 1 = [-0.57, 146.65, -750.06]
Point 2 = [-9.44, 147.22, -736.91]
Distance = 15.87
Euclidean Operation Count = 1000000
Execution Time = 1.2640199661254883 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [-9.44, 147.22, -736.91]
Point 2 = [-0.57, 146.65, -750.06]
Distance = 15.87
Euclidean Operation Count = 42441
Execution Time = 0.05299854278564453 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```



4.3.2 6D



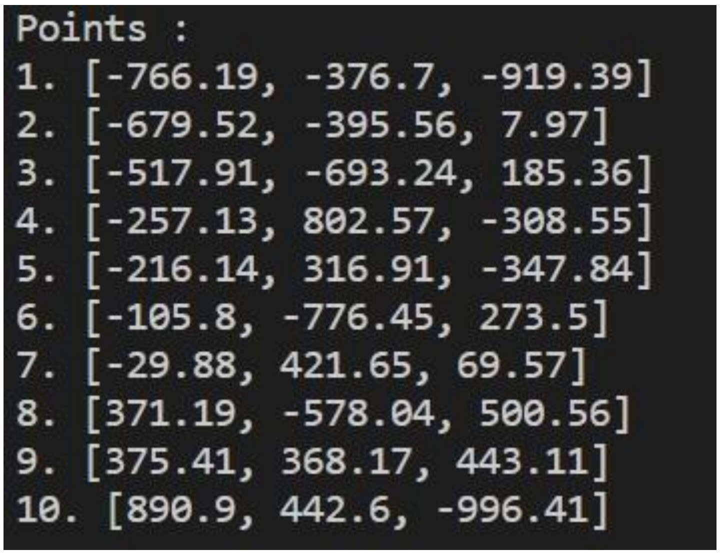
```
Dimension n (n > 0) : 6
Total Points k (k > 1) : 1000
```



```
Closest pair of points :
Brute Force :
Point 1 = [533.65, -644.33, 736.0, 543.13, -148.97, 240.06]
Point 2 = [532.57, -654.81, 734.27, 625.62, -58.28, 220.31]
Distance = 124.63
Euclidean Operation Count = 1000000
Execution Time = 1.961632251739502 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0

Divide and Conquer :
Point 1 = [532.57, -654.81, 734.27, 625.62, -58.28, 220.31]
Point 2 = [533.65, -644.33, 736.0, 543.13, -148.97, 240.06]
Distance = 124.63
Euclidean Operation Count = 265209
Execution Time = 0.4915761947631836 seconds
Run in AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD, Windows-10-10.0.19044-SP0
```

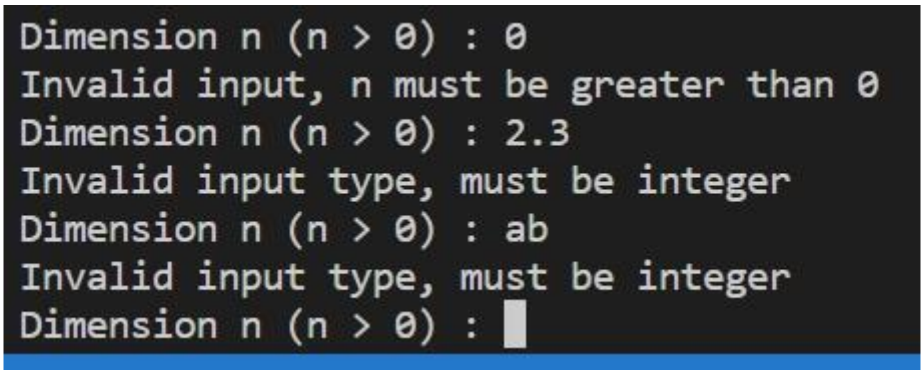
4.5 Contoh keluaran semua titik

Dengan sepuluh titik 3D, dihasilkan keluaran

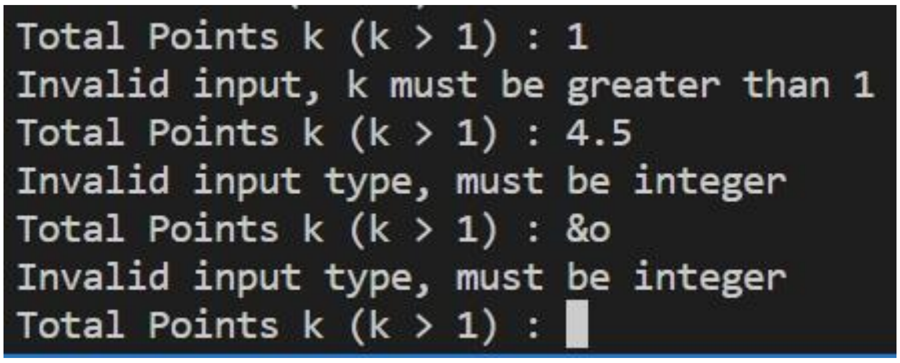


```
Points :
1. [-766.19, -376.7, -919.39]
2. [-679.52, -395.56, 7.97]
3. [-517.91, -693.24, 185.36]
4. [-257.13, 802.57, -308.55]
5. [-216.14, 316.91, -347.84]
6. [-105.8, -776.45, 273.5]
7. [-29.88, 421.65, 69.57]
8. [371.19, -578.04, 500.56]
9. [375.41, 368.17, 443.11]
10. [890.9, 442.6, -996.41]
```

4.6 Contoh apabila masukan tidak sesuai



```
Dimension n (n > 0) : 0
Invalid input, n must be greater than 0
Dimension n (n > 0) : 2.3
Invalid input type, must be integer
Dimension n (n > 0) : ab
Invalid input type, must be integer
Dimension n (n > 0) :
```



```
Total Points k (k > 1) : 1
Invalid input, k must be greater than 1
Total Points k (k > 1) : 4.5
Invalid input type, must be integer
Total Points k (k > 1) : &o
Invalid input type, must be integer
Total Points k (k > 1) :
```

LAMPIRAN

Link Repository Github:

https://github.com/rayhanp1402/Tucil2_13521112

Check List:

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil running	✓	
3. Program dapat menerima masukan 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 2)*. Diakses pada 27 Februari 2023 pukul 19.17 dari sumber https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf

matplotlib.org. *matplotlib.pyplot documentation*. Diakses pada 27 Februari 22.03 dari sumber https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html

iDeer7 (2021). *Closest Pair of Points (Divide and Conquer) Explained*. Diakses pada 27 Februari 21.13 dari sumber https://www.youtube.com/watch?v=6u_hWxbOc7E&t=398s