Tugas Kecil 2 IF2211 Strategi Algoritma “Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer*”

Disusun Oleh:

Bagas Aryo Seto	13521081
Farhan Nabil Suryono	13521114

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. Spesifikasi Tugas

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 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 sepasang titik yang jaraknya terdekat datu sama lain dengan menerapkan algoritma Divide and Conquer untuk penyelesaiannya, dan perbandingannya dengan Algoritma Brute Force.

2. Penjelasan Program

Divide and Conquer adalah salah satu strategi dalam membuat algoritma pemecahan masalah. Divide and Conquer pada intinya adalah dimulai dari proses *divide* yaitu memecah masalah menjadi masalah-masalah yang lebih kecil. Setelah itu, proses *conquer* dilakukan yaitu menyelesaikan masalah-masalah yang telah dipecah. Terakhir, proses *combine* adalah proses dimana masalah-masalah kecil yang telah dipecahkan digabung lagi sehingga didapat solusi dari masalah utama.

Implementasi program yang dibuat penulis berjalan dengan langkah-langkah sebagai berikut:

1. Program meminta masukan *user* berupa jumlah titik yang ingin dihasilkan dan jumlah dimensi dari titik tersebut
2. Program akan menghasilkan titik-titik unik sesuai permintaan *user* secara acak.
3. Program akan menghitung hasil secara *brute force* untuk selanjutnya akan digunakan untuk memeriksa hasil yang didapat melalui *divide and conquer*.
4. Setelah *brute force*, program akan menghitung hasil dengan *divide and conquer*.
5. *Divide and conquer* dimulai dengan melakukan pengurutan berdasarkan absis untuk memudahkan proses *divide*.
6. Selanjutnya, program akan memeriksa jumlah titik, bila jumlah titik sudah kurang dari sama dengan 3 maka program akan mengerjakan secara *brute force* sebagai *Base Case*.

7. Program akan membagi titik-titik menjadi 2 sisi berdasarkan absis dan memanggil diri secara rekursif dengan titik-titik yang telah terbagi.
8. Pada setiap rekursif, program akan menyelesaikan setiap sub-persoalan dan mencari jarak terkecil dari kedua sisi. Hasil jarak terkecil ini lalu dibandingkan dengan 2 titik dari sisi berbeda yang mungkin lebih pendek. Proses ini ditingkatkan efisiensinya dengan mengabaikan kombinasi dimana jarak pada suatu dimensi lebih dari sama dengan jarak terkecil yang sudah didapat.
9. Program akan terus menerus melakukan proses *combine* dan akan mengembalikan sepasang titik dan jaraknya.
10. Terakhir, program akan meng-output jarak 2 titik terdekat beserta titik-titiknya yang didapat melalui *brute force* dan *divide and conquer*. Selain itu, program juga memiliki opsi untuk melakukan visualisasi bila dimensi yang dipilih adalah 3 dimensi. Program juga akan meng-output durasi perhitungan dan berapa kali fungsi *euclidean distance* dipanggil.

3. Source Code Program

main.py

```
from point import Point
import randomGenerator
import bruteforce
import dnc
import visualization
import sort
import time
import fileReader


def main():
    print()
    print("======================================================================")
    print("                       Closest Pair Finder                            ")
    print("======================================================================")

    # # Input and validation
    # # (BONUS 2) Function Generalization for Multiple Dimensions

    inp = input("\nInput using file? (Y/N): ")
    if inp.lower().find("y") != -1:
        fname = input("Enter complete file path: ")
        try: listOfPoints = fileReader.readFile(fname)
        except:
            print("invalid file name or invalid file content!\n")
            return
    else:
        dim = int(input("Enter Point Dimension   : "))
        while dim < 2:
            print("Invalid Input! Dimension must be higher than 1.\n")
            dim = int(input("Enter Point Dimension   : "))

        num = int(input("Enter Number of Points  : "))
        while num < 2:
            print("Invalid input! There must be more than 1 point.\n")
            num = int(input("Enter Number of Points  : "))

        # Generate N random points
        listOfPoints = randomGenerator.generateRandomPoints(num, dim)
    # Sort by x values
    sort.quickSort(listOfPoints, 0, len(listOfPoints) - 1, key=lambda p:
p.getCoordinateValue(0))

    # Reset calculation counter
    bruteforce.euclidCntBF = 0
    dnc.euclidCntDnC = 0
    dnc.euclidCntDnCInBf = 0

    # Get the result using Brute Force Algorithm
    t = time.time()
    bfPairs, bfDist = bruteforce.closestPairBruteForce(listOfPoints)
    bfTime = time.time() - t

    # Get the result using Divide and Conquer Algorithm
    t = time.time()
    dncPairs, dncDist = dnc.closestPair(listOfPoints)
    dncTime = time.time() - t

    # Output result
    # totlen = max(27, max(len(str(dncPairs[0])), len(str(dncPairs[1])))) + 1

    pairPoints = list(set([points for pair in dncPairs for points in pair]))
    maxlen = max([len(str(ppoint)) for ppoint in pairPoints] + [27]) + 1
    pad = min(maxlen, 70)

    print("\n")
    print("                   | {: <{}}|  Divide and Conquer".format("Brute Force",
pad))
    print("-------------------+-{}+-{}".format("-" * pad, "-" * pad))

    n = len(dncPairs)
    for i in range(n):
        totlen = max(27, max(len(str(dncPairs[i][0])), len(str(dncPairs[i][1])))) +
1
        if totlen == pad:
            print("Pairs {: <{}} | {: <{}}| {}".format(i+1 if n>1 else " ", 12,
str(bfPairs[i][0]), pad, str(dncPairs[i][0])))
            print("                   | {: <{}}| {}".format(str(bfPairs[i][1]),
pad, str(dncPairs[i][1])))
        else:
            bf1, bf2 = str(bfPairs[i][0]), str(bfPairs[i][1])
            dc1, dc2 = str(dncPairs[i][0]), str(dncPairs[i][1])
            start1, start2, end1, end2 = 0, 0, 0, 0
            while end1 < len(bf1) or end2 < len(dc1):
                start1, start2 = end1, end2
                end1 = bf1.rfind(',',start1,start1+pad)+1 if len(bf1)-start1 > pad
else len(bf1)
                end2 = dc1.rfind(',',start2,start2+pad)+1 if len(dc1)-start2 > pad
else len(dc1)
                if start1 == 0 and start2 == 0:
                    print("Pairs {: <{}} | {: <{}}| {}".format(i+1 if n>1 else " ",
12, bf1[start1:end1], pad, dc1[start2:end2]))
                else:
                    print("                   | {: <{}}|
{}".format(bf1[start1:end1], pad, dc1[start2:end2]))
            start1, start2, end1, end2 = 0, 0, 0, 0
            while end1 < len(bf2) or end2 < len(dc2):
                start1, start2 = end1, end2
                end1 = bf2.rfind(',',start1,start1+pad)+1 if len(bf2)-start1 > pad
else len(bf2)
                end2 = dc2.rfind(',',start2,start2+pad)+1 if len(dc2)-start2 > pad
else len(dc2)
                print("                   | {: <{}}| {}".format(bf2[start1:end1],
pad, dc2[start2:end2]))
        print("-------------------+-{}+-{}".format("-" * pad, "-" * pad))

    print("Distance           | {: <{}}| {}".format(str(bfDist)[:pad - 1], pad,
str(dncDist)[:pad - 1]))
    print("-------------------+-{}+-{}".format("-" * pad, "-" * pad))
    print("Calculation Amount | {: <{}}| {}".format(str(int(bruteforce.euclidCntBF
- dnc.euclidCntDnCInBf))[:pad - 1],
                                                    pad, str(int(dnc.euclidCntDnC +
dnc.euclidCntDnCInBf))[:pad - 1]))
    print("-------------------+-{}+-{}".format("-" * pad, "-" * pad))
    print("Time Taken (s)     | {: <{}}| {}".format(str(bfTime)[:pad - 1], pad,
str(dncTime)[:pad - 1]))

    # (BONUS 1) Visualize for 3 Dimensional Input
    if listOfPoints[0].getDimension() == 3:
        vis = input("\nVisualize 3 Dimensional Plane? (Y/N) : ")
        if vis.lower().find("y") != -1:
            visualization.visualize3D(listOfPoints, dncPairs)
    if listOfPoints[0].getDimension() == 2:
        vis = input("\nVisualize 2 Dimensional Plane? (Y/N) : ")
        if vis.lower().find("y") != -1:
            visualization.visualize2D(listOfPoints, dncPairs)
```

```python
    print()


# Program Utama
if __name__ == "__main__":
    # main()
    while True:
        main()
        ex = input("Exit program? (Y/N) : ")
        if ex.lower().find("y") != -1:
            print("\nbye~ (^.^)/")
            break
```

point.py

```python
class Point:
    def __init__(self, coordinates):
        # Konstruktor
        self.coordinates = coordinates
        self.dimension = len(coordinates)

    def __repr__(self):
        # Representasi Titik
        s = "("
        for i in range(self.dimension):
            if i != 0: s += ", "
            s += str(self.coordinates[i])
        s += ")"
        return s

    def distanceBetween(self, other):
        # Menghitung Euclidean Distance Antara Titik self dan Titik other
        val = 0
        for i in range(self.dimension):
            val += (self.coordinates[i] - other.coordinates[i]) ** 2
        return val ** 0.5

    def getCoordinates(self):
        return self.coordinates

    def getCoordinateValue(self, index):
        return self.coordinates[index]

    def getDimension(self):
        return self.dimension
```

dnc.py

```
import bruteforce
import sort
import randomGenerator
from point import Point

euclidCntDnC = 0
euclidCntDnCInBf = 0


def closestPair(listOfPoints):
    # Euclidean counter
    global euclidCntDnC
    global euclidCntDnCInBf

    n = len(listOfPoints)

    # Base Case
    if n <= 3:
        euclidCntDnCInBf += (n * (n - 1)) / 2
        return bruteforce.closestPairBruteForce(listOfPoints)

    # Recursive Case
    # Get The Dimension
    dim = listOfPoints[0].getDimension()
    sqrtdim = dim ** 0.5

    # Get The Middle Points
    midIndex = n // 2

    # Get Divided Points
    leftPoints = listOfPoints[:midIndex]
    rightPoints = listOfPoints[midIndex:]

    # Get Closest Pair in Both Side
    leftPairs, leftDistance = closestPair(leftPoints)
    rightPairs, rightDistance = closestPair(rightPoints)

    # Get Minimum Distance from Both Closest Pair
    if leftDistance < rightDistance:
        bestPairs = leftPairs
    elif leftDistance > rightDistance:
        bestPairs = rightPairs
    else:
        bestPairs = leftPairs + rightPairs
    minDist = min(leftDistance, rightDistance)

    # Get the Absis Coordinate of the Center
    centerAbsis = (leftPoints[-1].getCoordinateValue(0) +
rightPoints[0].getCoordinateValue(0)) / 2

    # Filter the Points That Distance to Center Smaller than min_dist
    candidatePoints = []
    for p in listOfPoints:
        if abs(p.getCoordinateValue(0) - centerAbsis) <= minDist:
            candidatePoints += [p]

    # Sort by y value
    sort.quickSort(candidatePoints, 0, len(candidatePoints)-1, key=lambda p:
p.getCoordinateValue(1))

    # Merging Process
    for i in range(len(candidatePoints) - 1):
        for j in range(i + 1, len(candidatePoints)):
            if candidatePoints[j].getCoordinateValue(1) -
candidatePoints[i].getCoordinateValue(1) > minDist:
                break

            pos = True
            mhtDist = 0
            for k in range(dim):
                mhtDist += abs(candidatePoints[j].getCoordinateValue(k) -
candidatePoints[i].getCoordinateValue(k))
                if mhtDist > minDist * sqrtdim:
                    pos = False
                    break
            if not pos:
                continue

            temp = candidatePoints[i].distanceBetween(candidatePoints[j])
            euclidCntDnC += 1
            if temp < minDist:
                minDist = temp
                bestPairs = [[candidatePoints[i], candidatePoints[j]]]
            elif temp == minDist:
                addedPair = [candidatePoints[i], candidatePoints[j]]
                if addedPair not in bestPairs and [candidatePoints[j], 
candidatePoints[i]] not in bestPairs:
                    bestPairs += [addedPair]

    return bestPairs, minDist
```

randomGenerator.py

```
import random
from point import Point

# Konstanta Maksimal Nilai Koordinat
MAXN = 1000


def generateRandomPoints(num, dimension):
    # Menghasilkan Titik Random Sebanyak num Berdimensi dimension

    listPoint = []
    for i in range(num):
        coordinates = []
        for j in range(dimension):
            coordinates += [round(random.uniform(-MAXN, MAXN), 2)]
        listPoint += [Point(coordinates)]

    return listPoint
```

bruteforce.py

```
import randomGenerator

euclidCntBF = 0
```

```python
def closestPairBruteForce(listOfPoints):
    global euclidCntBF

    # Menentukan Closest Pair dengan Brute Force
    # Digunakan untuk Uji Coba
    closestDistance = -1
    closestPairs = []

    for i in range(len(listOfPoints) - 1):
        for j in range(i + 1, len(listOfPoints)):
            dist = listOfPoints[i].distanceBetween(listOfPoints[j])
            euclidCntBF += 1
            if closestDistance == -1 or dist < closestDistance:
                closestDistance = dist
                closestPairs = [[listOfPoints[i], listOfPoints[j]]]
            elif dist == closestDistance:
                closestPairs += [[listOfPoints[i], listOfPoints[j]]]

    return closestPairs, closestDistance
```

visualization.py

```python
import matplotlib.pyplot as plt
import randomGenerator
import bruteforce, dnc
from point import Point
import fileReader


def visualize3D(points, pairs):
    n = len(points)
    xAxis = []
    yAxis = []
    zAxis = []

    pairPoints = list(set([points for pair in pairs for points in pair]))
    labeled = [False for i in range(len(pairPoints))]

    for i in range(n):
        if points[i] in pairPoints: continue
        xAxis.append(points[i].coordinates[0])
        yAxis.append(points[i].coordinates[1])
        zAxis.append(points[i].coordinates[2])


    fig = plt.figure()
    ax = plt.axes(projection='3d')
    ax.set_title("3D VISUALIZATION")
    ax.scatter3D(xAxis, yAxis, zAxis, c="Blue", depthshade=False)

    p = len(pairs)
    for i in range(p):

        pair1 = pairs[i][0].getCoordinates()
        pair2 = pairs[i][1].getCoordinates()

        ax.scatter3D(pair1[0], pair1[1], pair1[2], c="Red")
        ax.scatter3D(pair2[0], pair2[1], pair2[2], c="Red")

        ax.plot((pair1[0], pair2[0]), (pair1[1], pair2[1]), (pair1[2], pair2[2]),
c="Red")

        if pair1[2] < pair2[2]: va1="top"; va2="bottom"
        else : va1="bottom"; va2="top"

        if not labeled[pairPoints.index(pairs[i][0])]:
            ax.text(pair1[0], pair1[1], pair1[2], "(%.1f,%.1f,%.1f)" % (pair1[0],
pair1[1], pair1[2]), size='small', va=va1, ha='center')
            labeled[pairPoints.index(pairs[i][0])] = True
        if not labeled[pairPoints.index(pairs[i][1])]:
            ax.text(pair2[0], pair2[1], pair2[2], "(%.1f,%.1f,%.1f)" % (pair2[0],
pair2[1], pair2[2]), size='small', va=va2, ha='center')
            labeled[pairPoints.index(pairs[i][1])] = True

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

def visualize2D(points, pairs):
    n = len(points)
    xAxis = []
    yAxis = []

    pairPoints = list(set([points for pair in pairs for points in pair]))
    labeled = [False for i in range(len(pairPoints))]
    for i in range(n):
        if points[i] in pairPoints: continue
        xAxis.append(points[i].coordinates[0])
        yAxis.append(points[i].coordinates[1])


    fig = plt.figure()
    ax = plt.axes()
    ax.set_title("2D VISUALIZATION")
    ax.scatter(xAxis, yAxis, c="Blue")

    p = len(pairs)
    for i in range(p):

        pair1 = pairs[i][0].getCoordinates()
        pair2 = pairs[i][1].getCoordinates()

        ax.scatter(pair1[0], pair1[1], c="Red")
        ax.scatter(pair2[0], pair2[1], c="Red")

        ax.plot((pair1[0], pair2[0]), (pair1[1], pair2[1]), c="Red")

        if pair1[1] > pair2[1]: va1="top"; va2="bottom"
        else : va1="bottom"; va2="top"

        if not labeled[pairPoints.index(pairs[i][0])]:
            ax.text(pair1[0], pair1[1], "(%.1f,%.1f)" % (pair1[0], pair1[1]),
size='small', va=va1, ha='center')
            labeled[pairPoints.index(pairs[i][0])] = True
        if not labeled[pairPoints.index(pairs[i][1])]:
            ax.text(pair2[0], pair2[1], "(%.1f,%.1f)" % (pair2[0], pair2[1]),
size='small', va=va2, ha='center')
            labeled[pairPoints.index(pairs[i][1])] = True

    ax.set_xlabel('X-Axis')
    ax.set_ylabel('Y-Axis')
    plt.grid()
    plt.show()
```

sort.py

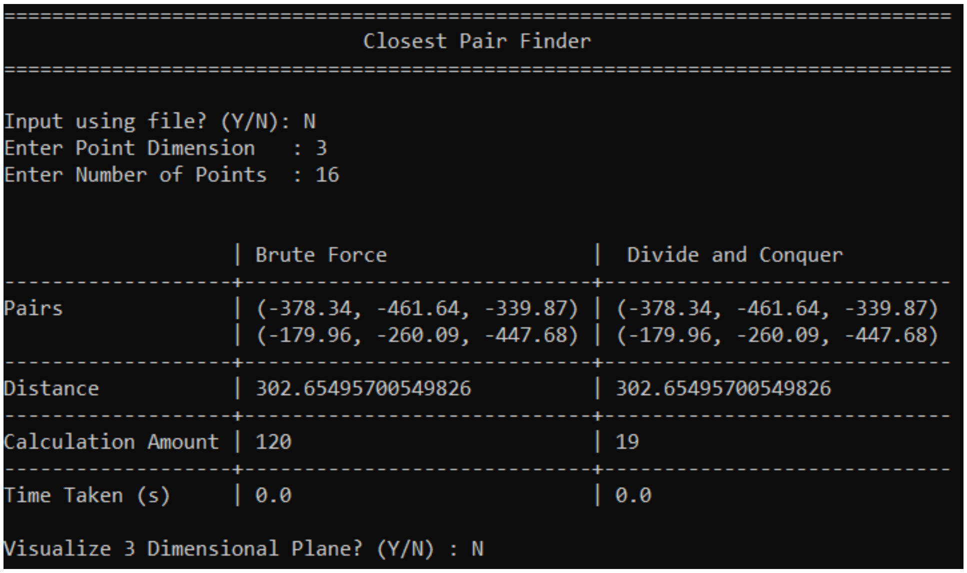
```
def partition(array, low, high, key):
    # choose the rightmost element as pivot
    pivot = array[high]

    # pointer for greater element
    i = low - 1

    # compare each element with pivot
    for j in range(low, high):
        if key(array[j]) <= key(pivot):
            i += 1
            array[i], array[j] = array[j], array[i]

    # Swap the pivot element with the greater element specified by i
    array[i + 1], array[high] = array[high], array[i + 1]

    # Return the position from where partition is done
    return i + 1

def quickSort(array, low, high, key=lambda x: x):
    if low < high:
        pi = partition(array, low, high, key)
        quickSort(array, low, pi - 1, key)
        quickSort(array, pi + 1, high, key)
```

fileReader.py

```
from point import Point


def readFile(fileName):
    f = open(fileName.encode('unicode_escape')
        .decode().replace("\"", ""), 'r')
    raw = f.readlines()

    for i in range(len(raw)):
        raw[i] = raw[i][:-1]

    dim = int(raw[0])
    num = int(raw[1])

    listOfPoints = []
    for i in range(num):
        p = [float(c) for c in raw[i + 2].split()]
        listOfPoints += [Point(p)]

    f.close()

    return listOfPoints
```

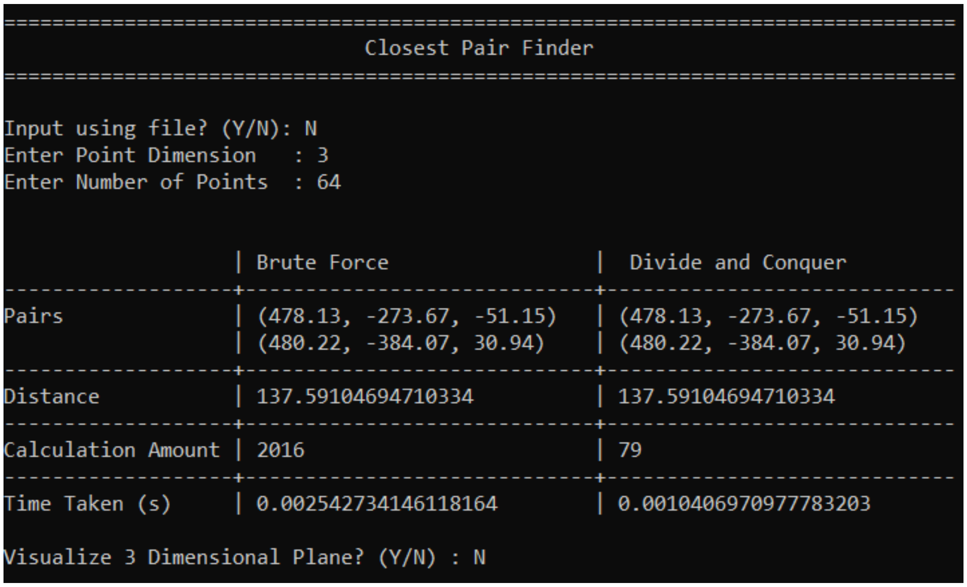
4. Screenshot Contoh I/O

- Testcase 1
  (dimensi : 3, *n* : 16, tanpa visualisasi)



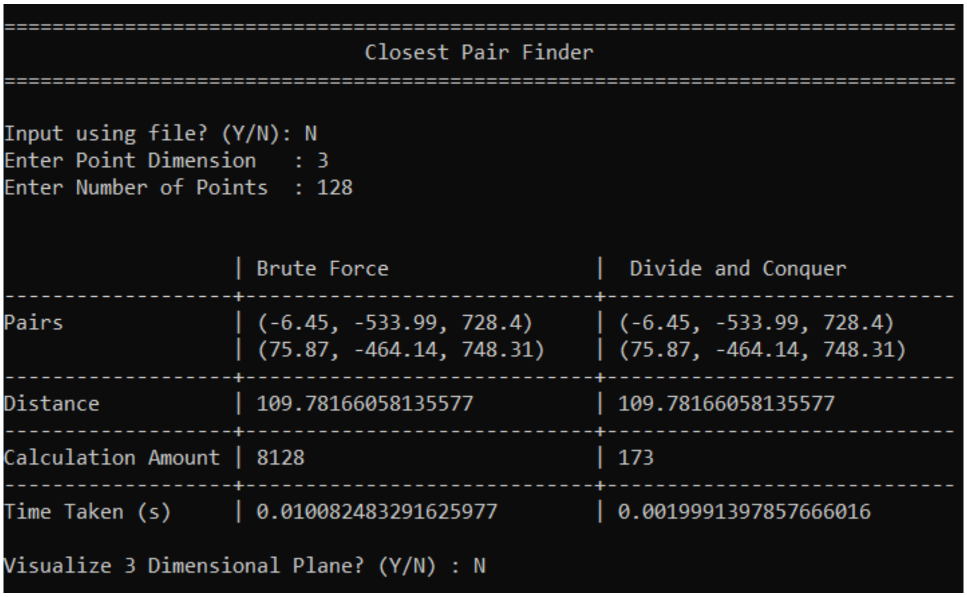
```
================================================================================
                              Closest Pair Finder
================================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 3
Enter Number of Points  : 16


                   | Brute Force                 |  Divide and Conquer
-------------------+-----------------------------+------------------------------
Pairs              | (-378.34, -461.64, -339.87) | (-378.34, -461.64, -339.87)
                   | (-179.96, -260.09, -447.68) | (-179.96, -260.09, -447.68)
-------------------+-----------------------------+------------------------------
Distance           | 302.65495700549826          | 302.65495700549826
-------------------+-----------------------------+------------------------------
Calculation Amount | 120                         | 19
-------------------+-----------------------------+------------------------------
Time Taken (s)     | 0.0                         | 0.0

Visualize 3 Dimensional Plane? (Y/N) : N
```

- Testcase 2
  (dimensi : 3, n : 64, tanpa visualisasi)



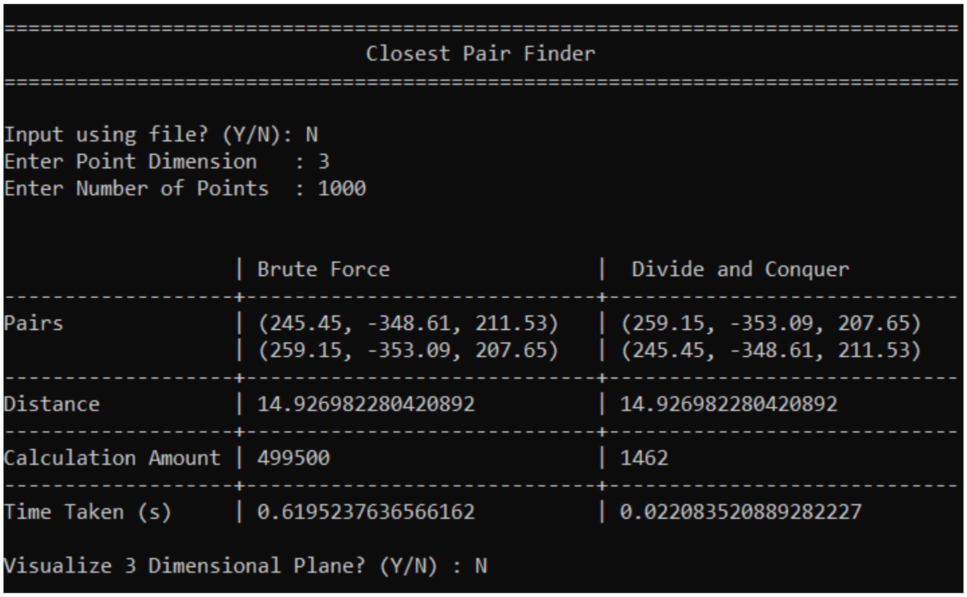
```
================================================================================
                              Closest Pair Finder
================================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 3
Enter Number of Points  : 64


                   | Brute Force                 |  Divide and Conquer
-------------------+-----------------------------+------------------------------
Pairs              | (478.13, -273.67, -51.15)   | (478.13, -273.67, -51.15)
                   | (480.22, -384.07, 30.94)    | (480.22, -384.07, 30.94)
-------------------+-----------------------------+------------------------------
Distance           | 137.59104694710334          | 137.59104694710334
-------------------+-----------------------------+------------------------------
Calculation Amount | 2016                        | 79
-------------------+-----------------------------+------------------------------
Time Taken (s)     | 0.002542734146118164        | 0.0010406970977783203

Visualize 3 Dimensional Plane? (Y/N) : N
```

- Testcase 3
  (dimensi : 3, n : 128, tanpa visualisasi)



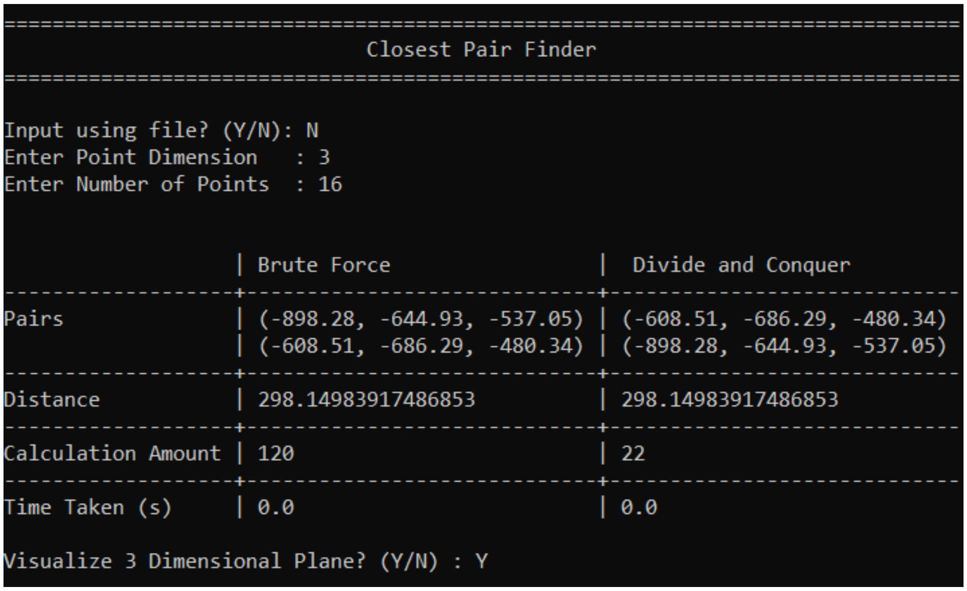
```
================================================================================
                              Closest Pair Finder
================================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 3
Enter Number of Points  : 128


                   | Brute Force              |  Divide and Conquer
-------------------+--------------------------+------------------------------
Pairs              | (-6.45, -533.99, 728.4)  | (-6.45, -533.99, 728.4)
                   | (75.87, -464.14, 748.31) | (75.87, -464.14, 748.31)
-------------------+--------------------------+------------------------------
Distance           | 109.78166058135577       | 109.78166058135577
-------------------+--------------------------+------------------------------
Calculation Amount | 8128                     | 173
-------------------+--------------------------+------------------------------
Time Taken (s)     | 0.010082483291625977     | 0.0019991397857666016

Visualize 3 Dimensional Plane? (Y/N) : N
```

- Testcase 4
  (dimensi : 3, n : 1000, tanpa visualisasi)

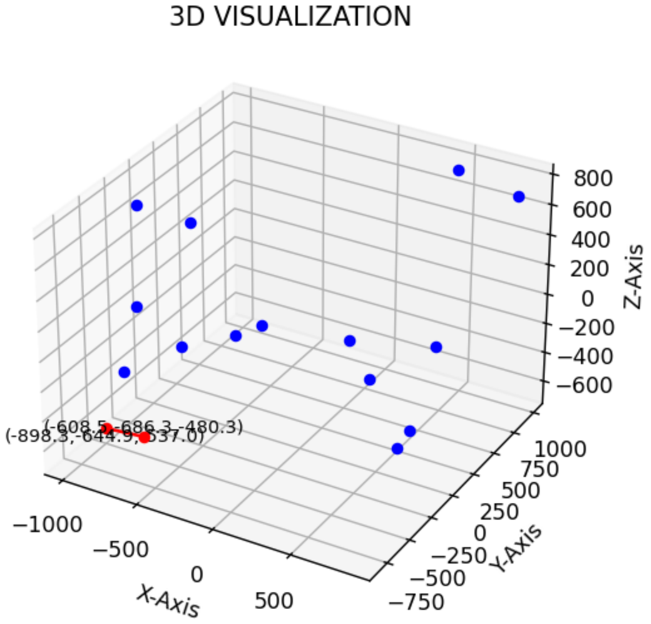


```
================================================================================
                              Closest Pair Finder
================================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 3
Enter Number of Points  : 1000


                   | Brute Force                |  Divide and Conquer
-------------------+----------------------------+------------------------------
Pairs              | (245.45, -348.61, 211.53)  | (259.15, -353.09, 207.65)
                   | (259.15, -353.09, 207.65)  | (245.45, -348.61, 211.53)
-------------------+----------------------------+------------------------------
Distance           | 14.926982280420892         | 14.926982280420892
-------------------+----------------------------+------------------------------
Calculation Amount | 499500                     | 1462
-------------------+----------------------------+------------------------------
Time Taken (s)     | 0.6195237636566162         | 0.022083520889282227

Visualize 3 Dimensional Plane? (Y/N) : N
```

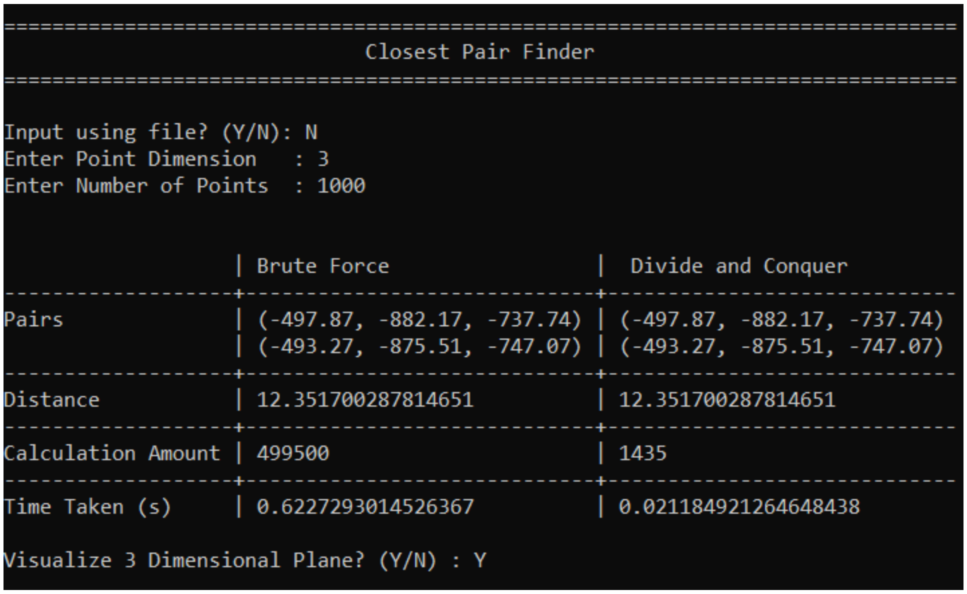
- Testcase 5
  (dimensi : 3, n : 16, dengan visualisasi)



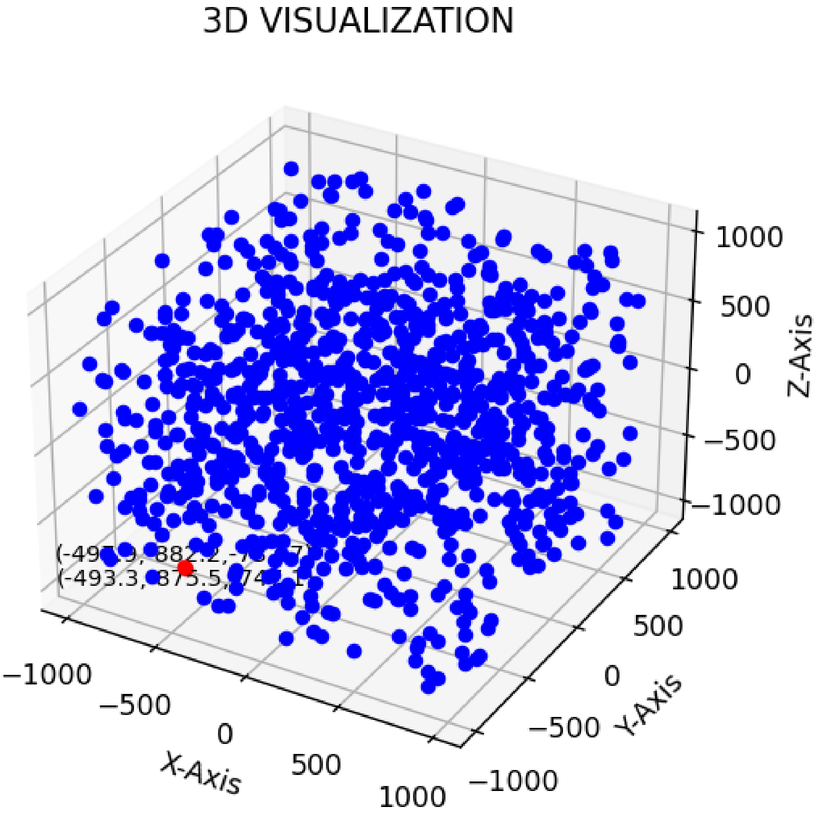
```
=================================================================================
                              Closest Pair Finder
=================================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 3
Enter Number of Points  : 16


                   | Brute Force                 |  Divide and Conquer
-------------------+-----------------------------+-----------------------------
Pairs              | (-898.28, -644.93, -537.05) | (-608.51, -686.29, -480.34)
                   | (-608.51, -686.29, -480.34) | (-898.28, -644.93, -537.05)
-------------------+-----------------------------+-----------------------------
Distance           | 298.14983917486853          | 298.14983917486853
-------------------+-----------------------------+-----------------------------
Calculation Amount | 120                         | 22
-------------------+-----------------------------+-----------------------------
Time Taken (s)     | 0.0                         | 0.0

Visualize 3 Dimensional Plane? (Y/N) : Y
```



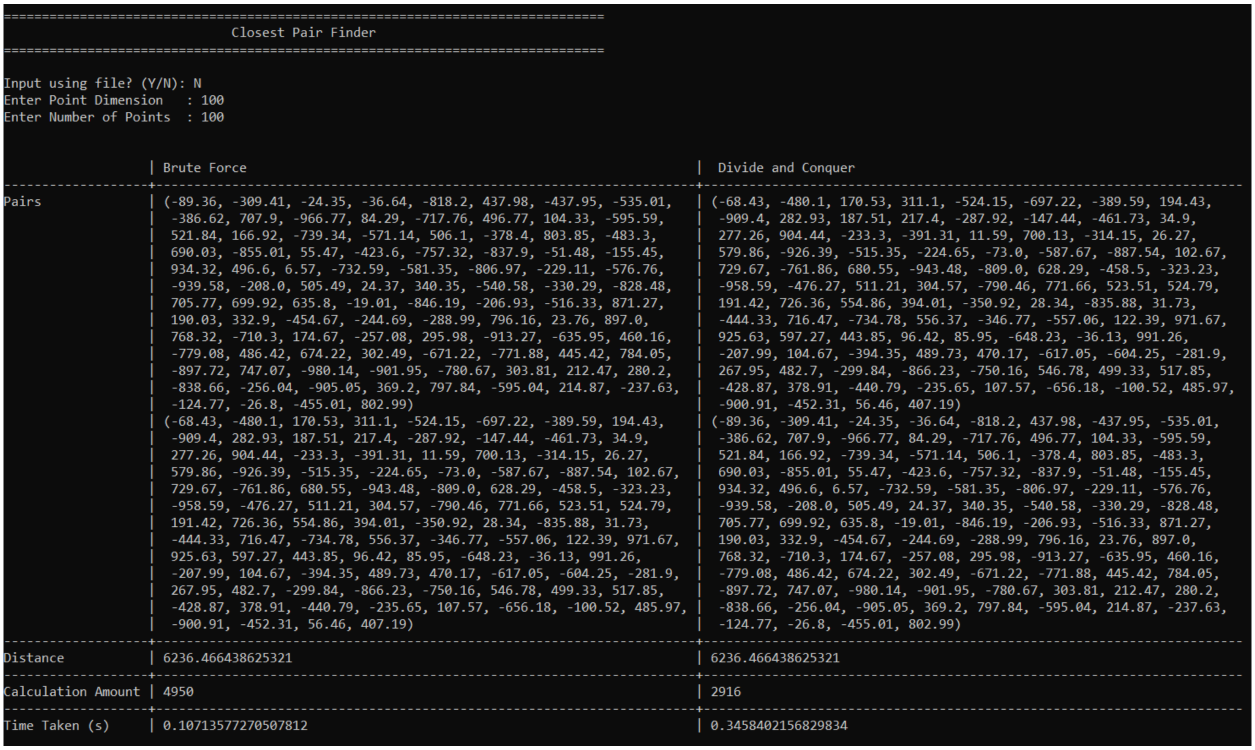
- Testcase 6
  (dimensi : 3, n : 1000, dengan visualisasi)



```
================================================================================
                              Closest Pair Finder
================================================================================

Input using file? (Y/N): N
Enter Point Dimension    : 3
Enter Number of Points   : 1000


                   | Brute Force                  |  Divide and Conquer
-------------------+------------------------------+-----------------------------
Pairs              | (-497.87, -882.17, -737.74) | (-497.87, -882.17, -737.74)
                   | (-493.27, -875.51, -747.07) | (-493.27, -875.51, -747.07)
-------------------+------------------------------+-----------------------------
Distance           | 12.351700287814651           | 12.351700287814651
-------------------+------------------------------+-----------------------------
Calculation Amount | 499500                       | 1435
-------------------+------------------------------+-----------------------------
Time Taken (s)     | 0.6227293014526367           | 0.021184921264648438

Visualize 3 Dimensional Plane? (Y/N) : Y
```

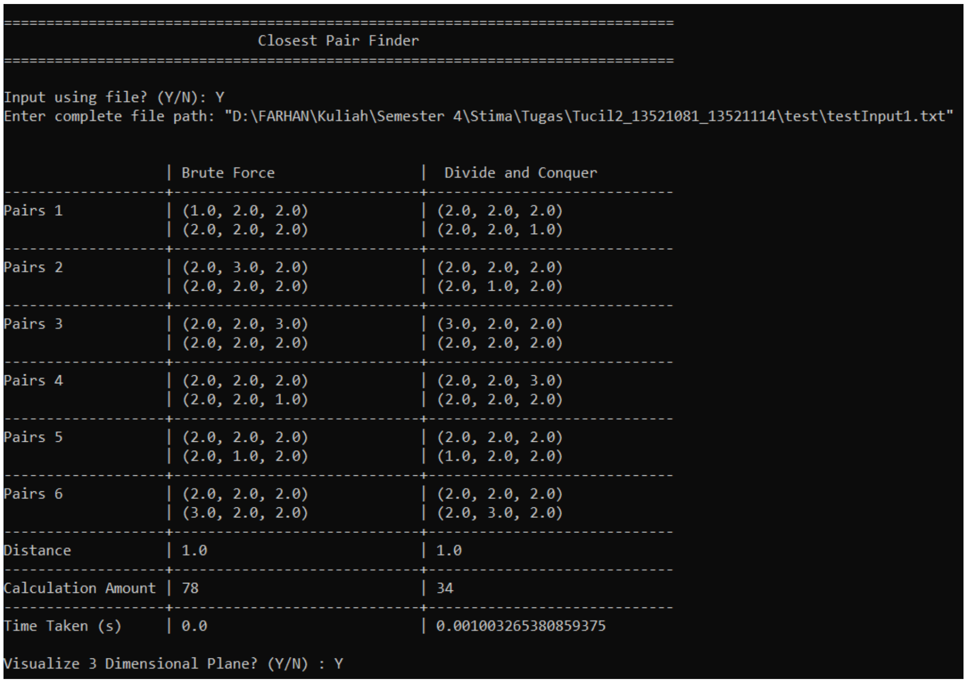


- Testcase 8
  (dimensi : 100, *n* : 100)

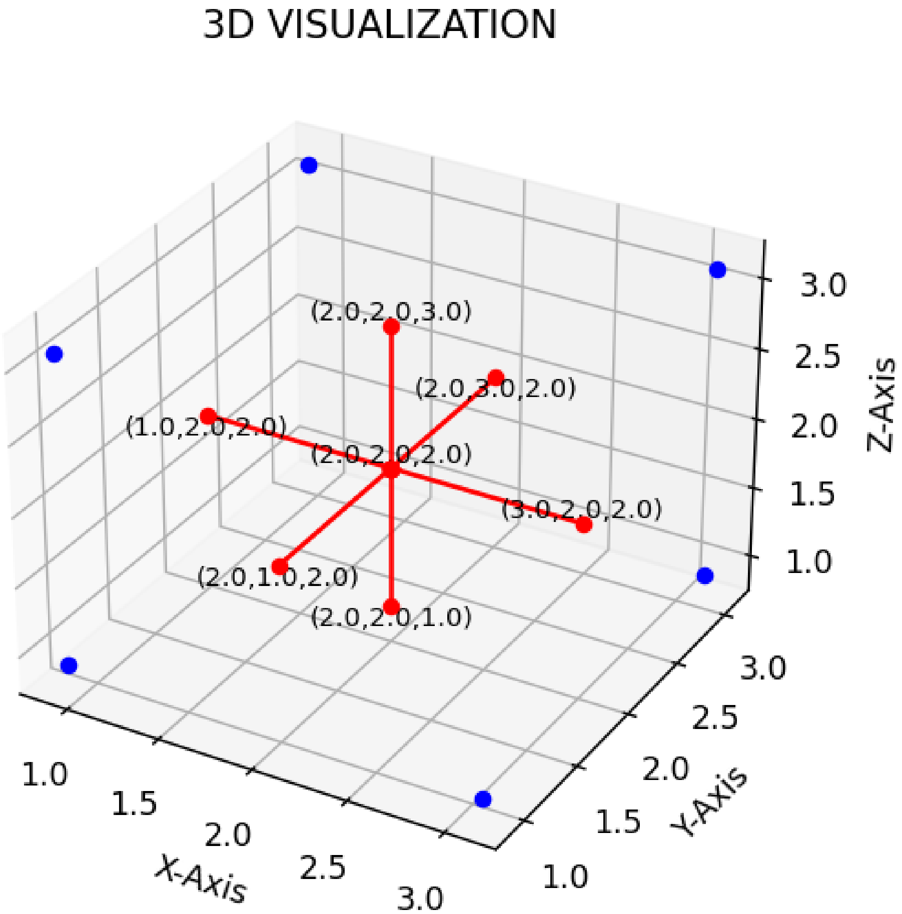


```
========================================================================
                           Closest Pair Finder
========================================================================

Input using file? (Y/N): N
Enter Point Dimension   : 100
Enter Number of Points  : 100


                  | Brute Force                                                        |  Divide and Conquer
------------------+--------------------------------------------------------------------+---------------------------------------------------------------------
Pairs             | (-89.36, -309.41, -24.35, -36.64, -818.2, 437.98, -437.95, -535.01, | (-68.43, -480.1, 170.53, 311.1, -524.15, -697.22, -389.59, 194.43,
                  |  -386.62, 707.9, -966.77, 84.29, -717.76, 496.77, 104.33, -595.59,  |  -909.4, 282.93, 187.51, 217.4, -287.92, -147.44, -461.73, 34.9,
                  |  521.84, 166.92, -739.34, -571.14, 506.1, -378.4, 803.85, -483.3,   |  277.26, 904.44, -233.3, -391.31, 11.59, 700.13, -314.15, 26.27,
                  |  690.03, -855.01, 55.47, -423.6, -757.32, -837.9, -51.48, -155.45,  |  579.86, -926.39, -515.35, -224.65, -73.0, -587.67, -887.54, 102.67,
                  |  934.32, 496.6, 6.57, -732.59, -581.35, -806.97, -229.11, -576.76,  |  729.67, -761.86, 680.55, -943.48, -809.0, 628.29, -458.5, -323.23,
                  |  -939.58, -208.0, 505.49, 24.37, 340.35, -540.58, -330.29, -828.48, |  -958.59, -476.27, 511.21, 304.57, -790.46, 771.66, 523.51, 524.79,
                  |  705.77, 699.92, 635.8, -19.01, -846.19, -206.93, -516.33, 871.27,  |  191.42, 726.36, 554.86, 394.01, -350.92, 28.34, -835.88, 31.73,
                  |  190.03, 332.9, -454.67, -244.69, -288.99, 796.16, 23.76, 897.0,    |  -444.33, 716.47, -734.78, 556.37, -346.77, -557.06, 122.39, 971.67,
                  |  768.32, -710.3, 174.67, -257.08, 295.98, -913.27, -635.95, 460.16, |  925.63, 597.27, 443.85, 96.42, 85.95, -648.23, -36.13, 991.26,
                  |  -779.08, 486.42, 674.22, 302.49, -671.22, -771.88, 445.42, 784.05, |  -207.99, 104.67, -394.35, 489.73, 470.17, -617.05, -604.25, -281.9,
                  |  -897.72, 747.07, -980.14, -901.95, -780.67, 303.81, 212.47, 280.2, |  267.95, 482.7, -299.84, -866.23, -750.16, 546.78, 499.33, 517.85,
                  |  -838.66, -256.04, -905.05, 369.2, 797.84, -595.04, 214.87, -237.63, |  -428.87, 378.91, -440.79, -235.65, 107.57, -656.18, -100.52, 485.97,
                  |  -124.77, -26.8, -455.01, 802.99)                                   |  -900.91, -452.31, 56.46, 407.19)
                  | (-68.43, -480.1, 170.53, 311.1, -524.15, -697.22, -389.59, 194.43, | (-89.36, -309.41, -24.35, -36.64, -818.2, 437.98, -437.95, -535.01,
                  |  -909.4, 282.93, 187.51, 217.4, -287.92, -147.44, -461.73, 34.9,    |  -386.62, 707.9, -966.77, 84.29, -717.76, 496.77, 104.33, -595.59,
                  |  277.26, 904.44, -233.3, -391.31, 11.59, 700.13, -314.15, 26.27,    |  521.84, 166.92, -739.34, -571.14, 506.1, -378.4, 803.85, -483.3,
                  |  579.86, -926.39, -515.35, -224.65, -73.0, -587.67, -887.54, 102.67, |  690.03, -855.01, 55.47, -423.6, -757.32, -837.9, -51.48, -155.45,
                  |  729.67, -761.86, 680.55, -943.48, -809.0, 628.29, -458.5, -323.23, |  934.32, 496.6, 6.57, -732.59, -581.35, -806.97, -229.11, -576.76,
                  |  -958.59, -476.27, 511.21, 304.57, -790.46, 771.66, 523.51, 524.79, |  -939.58, -208.0, 505.49, 24.37, 340.35, -540.58, -330.29, -828.48,
                  |  191.42, 726.36, 554.86, 394.01, -350.92, 28.34, -835.88, 31.73,    |  705.77, 699.92, 635.8, -19.01, -846.19, -206.93, -516.33, 871.27,
                  |  -444.33, 716.47, -734.78, 556.37, -346.77, -557.06, 122.39, 971.67, |  190.03, 332.9, -454.67, -244.69, -288.99, 796.16, 23.76, 897.0,
                  |  925.63, 597.27, 443.85, 96.42, 85.95, -648.23, -36.13, 991.26,     |  768.32, -710.3, 174.67, -257.08, 295.98, -913.27, -635.95, 460.16,
                  |  -207.99, 104.67, -394.35, 489.73, 470.17, -617.05, -604.25, -281.9, |  -779.08, 486.42, 674.22, 302.49, -671.22, -771.88, 445.42, 784.05,
                  |  267.95, 482.7, -299.84, -866.23, -750.16, 546.78, 499.33, 517.85,  |  -897.72, 747.07, -980.14, -901.95, -780.67, 303.81, 212.47, 280.2,
                  |  -428.87, 378.91, -440.79, -235.65, 107.57, -656.18, -100.52, 485.97, |  -838.66, -256.04, -905.05, 369.2, 797.84, -595.04, 214.87, -237.63,
                  |  -900.91, -452.31, 56.46, 407.19)                                   |  -124.77, -26.8, -455.01, 802.99)
------------------+--------------------------------------------------------------------+---------------------------------------------------------------------
Distance          | 6236.466438625321                                                  | 6236.466438625321
------------------+--------------------------------------------------------------------+---------------------------------------------------------------------
Calculation Amount | 4950                                                              | 2916
------------------+--------------------------------------------------------------------+---------------------------------------------------------------------
Time Taken (s)    | 0.10713577270507812                                                | 0.3458402156829834
```

- Testcase 9 : Input dari File yaitu file testInput1.txt pada folder test (dengan visualisasi)



```
==============================================================================
                              Closest Pair Finder
==============================================================================

Input using file? (Y/N): Y
Enter complete file path: "D:\FARHAN\Kuliah\Semester 4\Stima\Tugas\Tucil2_13521081_13521114\test\testInput1.txt"


                   | Brute Force                  |  Divide and Conquer
-------------------+------------------------------+-----------------------------
Pairs 1            | (1.0, 2.0, 2.0)              | (2.0, 2.0, 2.0)
                   | (2.0, 2.0, 2.0)              | (2.0, 2.0, 1.0)
-------------------+------------------------------+-----------------------------
Pairs 2            | (2.0, 3.0, 2.0)              | (2.0, 2.0, 2.0)
                   | (2.0, 2.0, 2.0)              | (2.0, 1.0, 2.0)
-------------------+------------------------------+-----------------------------
Pairs 3            | (2.0, 2.0, 3.0)              | (3.0, 2.0, 2.0)
                   | (2.0, 2.0, 2.0)              | (2.0, 2.0, 2.0)
-------------------+------------------------------+-----------------------------
Pairs 4            | (2.0, 2.0, 2.0)              | (2.0, 2.0, 3.0)
                   | (2.0, 2.0, 1.0)              | (2.0, 2.0, 2.0)
-------------------+------------------------------+-----------------------------
Pairs 5            | (2.0, 2.0, 2.0)              | (2.0, 2.0, 2.0)
                   | (2.0, 1.0, 2.0)              | (1.0, 2.0, 2.0)
-------------------+------------------------------+-----------------------------
Pairs 6            | (2.0, 2.0, 2.0)              | (2.0, 2.0, 2.0)
                   | (3.0, 2.0, 2.0)              | (2.0, 3.0, 2.0)
-------------------+------------------------------+-----------------------------
Distance           | 1.0                          | 1.0
-------------------+------------------------------+-----------------------------
Calculation Amount | 78                           | 34
-------------------+------------------------------+-----------------------------
Time Taken (s)     | 0.0                          | 0.001003265380859375

Visualize 3 Dimensional Plane? (Y/N) : Y
```


3D VISUALIZATION
(2.0,2.0,3.0)
(2.0,3.0,2.0)
(1.0,2.0,2.0)
(2.0,2.0,2.0)
(3.0,2.0,2.0)
(2.0,1.0,2.0)
(2.0,2.0,1.0)
X-Axis
1.0
1.5
2.0
2.5
3.0
Y-Axis
1.0
1.5
2.0
2.5
3.0
Z-Axis
1.0
1.5
2.0
2.5
3.0

5. Lampiran

Link Repository Implementasi Program:
https://github.com/Altair1618/Tucil2_13521081_13521114

6. Tabel Checklist

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	✓	