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

# Program Utama
if __name__ == "__main__":
    print()
    print("============================================================")
    print("                    Closest Pair Finder                     ")
    print("============================================================")

    # Input and validation
    # (BONUS 2) Function Generalization for Multiple Dimensions
    dim = int(input("\nEnter 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, num-1, key=lambda p:
p.getCoordinateValue(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
    pad = max(18, max(len(str(dncPairs[0])), len(str(dncPairs[1])))) + 1
    print("\n")
    print("                   | {: <{}}|  Divide and
Conquer".format("Brute Force", pad))
    print("-------------------+-{}+-{}".format("-"*pad, "-"*pad))
    print("Pairs              | {: <{}}| {}".format(str(bfPairs[0]), pad,
str(dncPairs[0])))
    print("                   | {: <{}}| {}".format(str(bfPairs[1]), pad,
str(dncPairs[1])))
    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(bruteforce.euclidCntBF)[:pad-1], pad,
str(dnc.euclidCntDnC)[:pad-1]))
    print("-------------------+-{}+-{}".format("-"*pad, "-"*pad))
    print("Time Taken         | {: <{}}| {}".format(str(bfTime)[:pad-1],
pad, str(dncTime)[:pad-1]))

    # (BONUS 1) Visualize for 3 Dimensional Input
    if dim == 3:
        vis = input("\nVisualize 3 Dimensional Plane? (Y/y) : ")
        if vis == 'y' or vis == 'Y':
            visualization.visualize3D(listOfPoints, dncPairs)
```

point.py

```
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
```

```
euclidCntDnC = 0

def closestPair(listOfPoints):
    # Base Case
    if len(listOfPoints) <= 3:
        return bruteforce.closestPairBruteForce(listOfPoints)

    # Euclidean counter
    global euclidCntDnC

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

    # Get The Middle Points
    midIndex = len(listOfPoints) // 2

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

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

    # Get Minimum Distance from Both Closest Pair
    bestPair = leftPair if leftDistance < rightDistance else rightPair
    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
            for k in range(2, dim):
                if abs(candidatePoints[j].getCoordinateValue(k) -
candidatePoints[i].getCoordinateValue(k)) >= minDist:
                    pos = False
                    break
            if not pos:
                continue

            temp = candidatePoints[i].distanceBetween(candidatePoints[j])
            euclidCntDnC += 1
            if temp < minDist:
                minDist = temp
                bestPair = [candidatePoints[i], candidatePoints[j]]

    return bestPair, 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
    # Dipastikan Semua Titik Unik
    listPoint = []
    for i in range(num):
        while True:
            coordinates = []
            for j in range(dimension):
                coordinates += [round(random.uniform(-MAXN, MAXN), 2)]
```

```
            if Point not in listPoint:
                break
        listPoint += [Point(coordinates)]

    return listPoint
```

bruteforce.py

```
euclidCntBF = 0

def closestPairBruteForce(listOfPoints):
    # Menentukan Closest Pair dengan Brute Force
    # Digunakan untuk Uji Coba
    closestDistance = -1
    indexPoint1 = -1
    indexPoint2 = -1
    global euclidCntBF
    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
                indexPoint1 = i
                indexPoint2 = j
    return [listOfPoints[indexPoint1], listOfPoints[indexPoint2]], closestDistance
```

visualization.py

```
import matplotlib.pyplot as plt

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

    for i in range(n):
        if points[i] == pair[0] or points[i] == pair[1]: 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.scatter3D(xAxis, yAxis, zAxis, c="Blue", depthshade=False)

    pair1 = pair[0].getCoordinates()
    pair2 = pair[1].getCoordinates()

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

    ax.text(pair1[0], pair1[1], pair1[2], "(%d,%d,%d)" % (pair1[0], pair1[1],
pair1[2]), size='small', va='bottom', ha='center')
    ax.text(pair2[0], pair2[1], pair2[2], "(%d,%d,%d)" % (pair2[0], pair2[1],
pair2[2]), size='small', va='top', ha='center')

    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    plt.show()


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

    for i in range(n):
        if points[i] == pair[0] or points[i] == pair[1]: continue
        xAxis.append(points[i].coordinates[0])
        yAxis.append(points[i].coordinates[1])

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

    pair1 = pair[0].getCoordinates()
    pair2 = pair[1].getCoordinates()

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

```
    ax.text(pair1[0], pair1[1], "(%d,%d)" % (pair1[0], pair1[1]), size='small',
va='bottom', ha='center')
    ax.text(pair2[0], pair2[1], "(%d,%d)" % (pair2[0], pair2[1]), size='small',
va='top', ha='center')

    ax.set_xlabel('x')
    ax.set_ylabel('y')
    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)
```

4. Screenshot Contoh I/O

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

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

Enter Point Dimension   : 3
Enter Number of Points  : 16


                  | Brute Force                 |  Divide and Conquer
------------------+-----------------------------+-----------------------------
Pairs             | (247.78, 244.08, -808.18)   | (412.91, -154.99, -773.53)
                  | (412.91, -154.99, -773.53)  | (247.78, 244.08, -808.18)
------------------+-----------------------------+-----------------------------
Distance          | 433.27289818311976          | 433.27289818311976
------------------+-----------------------------+-----------------------------
Calculation Amount | 128                        | 10
------------------+-----------------------------+-----------------------------
Time Taken        | 0.002002239227294922        | 0.0

Visualize 3 Dimensional Plane? (Y/y) : n
```

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

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

Enter Point Dimension   : 3
Enter Number of Points  : 64


                  | Brute Force                 |  Divide and Conquer
------------------+-----------------------------+-----------------------------
Pairs             | (78.29, -337.73, -128.05)   | (88.43, -412.29, -103.77)
                  | (88.43, -412.29, -103.77)   | (78.29, -337.73, -128.05)
------------------+-----------------------------+-----------------------------
Distance          | 79.06662759976551           | 79.06662759976551
------------------+-----------------------------+-----------------------------
Calculation Amount | 2048                       | 49
------------------+-----------------------------+-----------------------------
Time Taken        | 0.0200045108795166          | 0.0009877681732177734

Visualize 3 Dimensional Plane? (Y/y) : n
```

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

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

Enter Point Dimension   : 3
Enter Number of Points  : 128


                  | Brute Force                 |  Divide and Conquer
------------------+-----------------------------+-----------------------------
Pairs             | (873.63, -825.96, 295.93)   | (873.63, -825.96, 295.93)
                  | (925.63, -806.35, 298.74)   | (925.63, -806.35, 298.74)
------------------+-----------------------------+-----------------------------
Distance          | 55.645738381299246          | 55.645738381299246
------------------+-----------------------------+-----------------------------
Calculation Amount | 8192                       | 102
------------------+-----------------------------+-----------------------------
Time Taken        | 0.037989139556884766        | 0.003009319305419922

Visualize 3 Dimensional Plane? (Y/y) : n
```

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

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

Enter Point Dimension   : 3
Enter Number of Points  : 1000


                  | Brute Force                 |  Divide and Conquer
------------------+-----------------------------+-----------------------------
Pairs             | (516.99, -371.35, -869.41)  | (516.99, -371.35, -869.41)
                  | (528.6, -370.48, -880.05)   | (528.6, -370.48, -880.05)
------------------+-----------------------------+-----------------------------
Distance          | 15.772082931559801          | 15.772082931559801
------------------+-----------------------------+-----------------------------
Calculation Amount | 500036                     | 853
------------------+-----------------------------+-----------------------------
Time Taken        | 1.9295125007629395          | 0.029999494552612305

Visualize 3 Dimensional Plane? (Y/y) : n
```

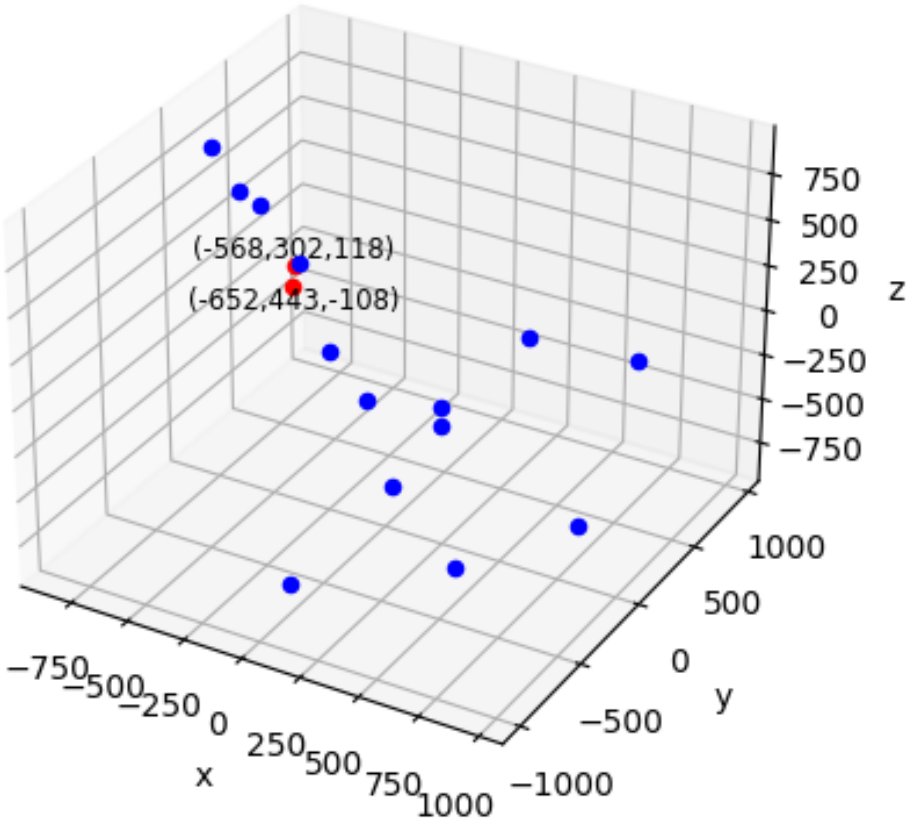
- Testcase 5
  (dimensi : 3, n : 16, dengan visualisasi)

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

Enter Point Dimension   : 3
Enter Number of Points  : 16


                  | Brute Force                 |  Divide and Conquer
------------------+-----------------------------+-----------------------------
Pairs             | (-652.65, 443.43, -108.93)  | (-568.52, 302.88, 118.11)
                  | (-568.52, 302.88, 118.11)   | (-652.65, 443.43, -108.93)
------------------+-----------------------------+-----------------------------
Distance          | 279.9630707789869           | 279.9630707789869
------------------+-----------------------------+-----------------------------
Calculation Amount | 128                        | 15
------------------+-----------------------------+-----------------------------
Time Taken        | 0.013031482696533203        | 0.0

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



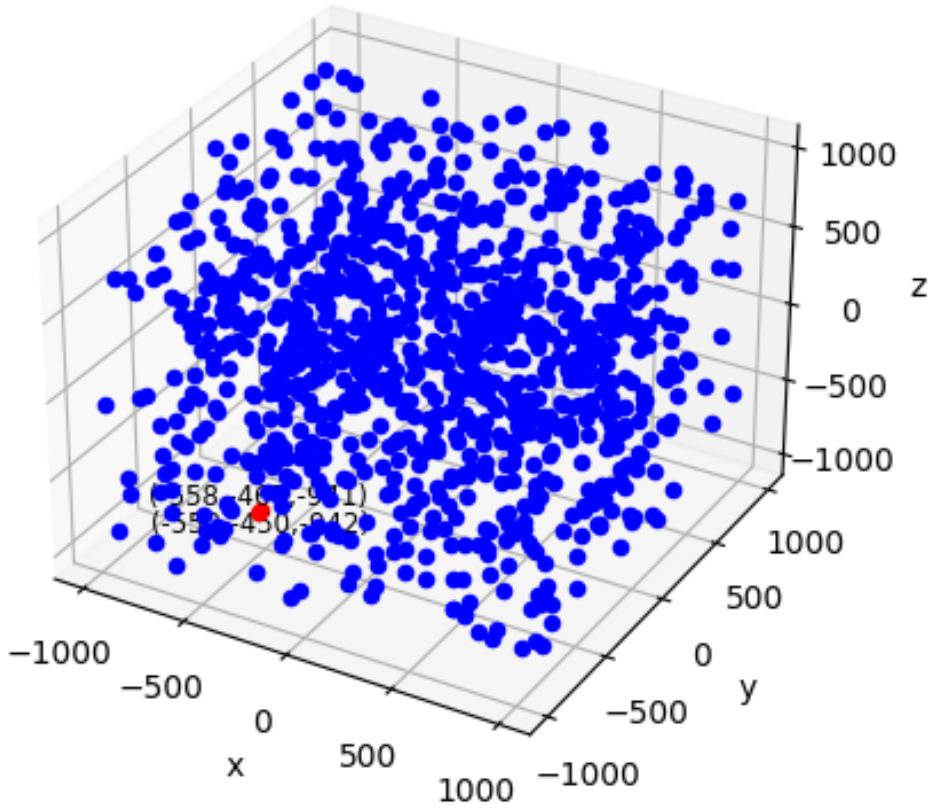
- Testcase 6
  (dimensi : 3, n : 1000, dengan visualisasi)

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

Enter Point Dimension   : 3
Enter Number of Points  : 1000


                  | Brute Force                | Divide and Conquer
------------------+----------------------------+----------------------------
Pairs             | (-558.13, -467.78, -941.26) | (-558.13, -467.78, -941.26)
                  | (-552.06, -450.85, -942.89) | (-552.06, -450.85, -942.89)
------------------+----------------------------+----------------------------
Distance          | 18.05897837641983          | 18.05897837641983
------------------+----------------------------+----------------------------
Calculation Amount | 500036                    | 839
------------------+----------------------------+----------------------------
Time Taken        | 1.0859999656677246         | 0.03153491020202637

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



- Testcase 7
  (dimensi : 4, *n* : 1000)

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

Enter Point Dimension   : 4
Enter Number of Points  : 1000


                   | Brute Force                          |  Divide and Conquer
-------------------+--------------------------------------+-------------------------------------
Pairs              | (234.72, -585.66, -905.75, 565.14) | (238.66, -593.08, -902.01, 577.01)
                   | (238.66, -593.08, -902.01, 577.01) | (234.72, -585.66, -905.75, 565.14)
-------------------+--------------------------------------+-------------------------------------
Distance           | 15.015475350450988                   | 15.015475350450988
-------------------+--------------------------------------+-------------------------------------
Calculation Amount | 500036                               | 1148
-------------------+--------------------------------------+-------------------------------------
Time Taken         | 1.5460052490234375                   | 0.0449976921081543
```

- Testcase 8
  (dimensi : 10, *n* : 1000)

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

Enter Point Dimension   : 10
Enter Number of Points  : 1000


                   | Brute Force                                                                  |  Divide and Conquer
-------------------+------------------------------------------------------------------------------+------------------------------------------------------------------------------
Pairs              | (26.71, -757.62, 583.94, -117.8, 271.82, -766.4, -54.77, 485.61, -332.06, 184.79) | (26.71, -757.62, 583.94, -117.8, 271.82, -766.4, -54.77, 485.61, -332.06, 184.79)
                   | (226.48, -511.01, 508.3, -175.58, 556.41, -660.18, 1.22, 441.5, -463.62, -36.17) | (226.48, -511.01, 508.3, -175.58, 556.41, -660.18, 1.22, 441.5, -463.62, -36.17)
-------------------+------------------------------------------------------------------------------+------------------------------------------------------------------------------
Distance           | 522.7528736410733                                                            | 522.7528736410733
-------------------+------------------------------------------------------------------------------+------------------------------------------------------------------------------
Calculation Amount | 500036                                                                       | 8715
-------------------+------------------------------------------------------------------------------+------------------------------------------------------------------------------
Time Taken         | 2.8710005283355713                                                           | 0.554999828338623
```

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	✓	