

Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

Pencarian Closest Pair Dengan Menggunakan
Algoritma Divide and Conquer



Disusun oleh:

Muhammad Habibi Husni

13521169

K01

**PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK
ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI
BANDUNG 2023**

1. Spesifikasi masalah

Pada permasalahan *closest pair* klasik, diberikan himpunan titik, P , yang terdiri dari n buah titik pada bidang 2-D, (x_i, y_i) , $i = 1, 2, 3, \dots, n$. Tentukan sepasang titik dengan jarak antar satu sama lain terdekat di dalam P . Permasalahan ini dapat diselesaikan dengan algoritma divide and conquer yang memiliki kompleksitas $O(n \log n)$ maupun brute force dengan kompleksitas $O(n^2)$. Solusi tersebut dapat diperluas untuk menyelesaikan permasalahan pada ruang vektor R^N untuk sembarang N positif. Ruang vektor R^N adalah sebuah ruang yang titik-titiknya direpresentasikan dengan nilai $(x_1, x_2, x_3, \dots, x_N)$. Dengan memperumum algoritma, maka permasalahan pada bidang 2D dan ruang 3D juga dapat diselesaikan dengan mudah. Untuk dua buah titik $P(p_1, p_2, \dots, p_N)$ dan $Q(q_1, q_2, \dots, q_N)$ pada ruang vektor R^N , jarak kedua titik tersebut dapat dihitung dengan rumus euclidean distance sebagai berikut :

$$\text{dist}(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_N - q_N)^2}$$

2. Algoritma Brute Force

Brute force merupakan salah satu penyelesaian secara *straightforward*. Dalam kasus permasalahan ini, solusi brute force dapat diaplikasikan dengan mengiterasi segala kombinasi titik yang ada pada ruang. Dengan melakukan *exhaustive search*, maka solusi brute force memiliki kompleksitas $O(n^2)$.

3. Algoritma Divide and Conquer

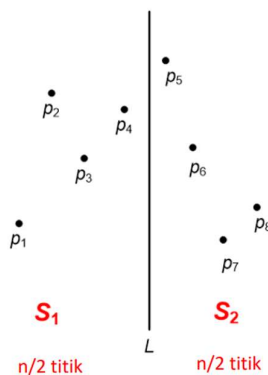
Algoritma divide and conquer memiliki tiga tahap yaitu : *solve*, *divide*, *conquer*, dan, *combine*. Dalam penyelesaian *closest pair* dalam program ini, berikut adalah pendefinisian algoritma tiap tahap tersebut:

1. Solve

Jika jumlah titik = 2, maka dapat dilakukan perhitungan jarak euclidean secara langsung. Namun jika jumlah titik = 1, maka nilai *closestPair* nya tidak dapat didefinisikan atau dapat dimisalkan dengan tak hingga.

2. Divide

Dalam tahap divide, himpunan titik dibagi menjadi dua buah himpunan titik, S_1 dan S_2 , yang jumlah titiknya sama. Pembagian titik dilakukan dengan memberikan *hyperplane* L yang sejajar pada axis-1 yang membagi dua daerah dengan banyak titik yang sama ($n/2$).

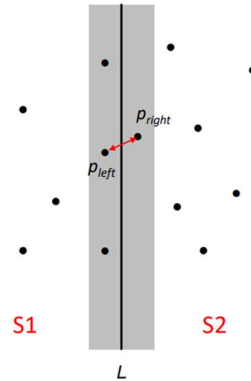


3. Conquer

Secara rekursif, terapkan algoritma DnC yang sama pada masing-masing bagian kiri dan kanan untuk memperoleh *closest pairs* pada S_1 dan S_2 , dimisalkan dengan $dist1$ dan $dist2$.

4. Combine

Gabungkan solusi $dist1$ dan $dist2$ menjadi solusi global $minDist$. Namun, masih terdapat kemungkinan terdapat pasangan titik terdekat yang salah satunya terletak di S_1 dan yang lainnya terletak di S_2 .



Maka perlu dilakukan pencarian pasangan titik terdekat yang axis-1 nya memiliki jarak $\leq minDist$ dari L . Agar pencarian dapat dilakukan dengan efisien, maka seluruh titik pada strip diproyeksikan pada salah satu sumbu-i pada ruang R^N . Setelah itu, dilakukan pengecekan masing-masing titik dengan titik yang jarak proyeksinya $\leq minDist$. Agar pengecekan dapat dilakukan dengan efisien, maka titik-titik tersebut di *presort* berdasarkan nilai pada sumbu-i nya dan lalu dilakukan iterasi dari nilai sumbu-i terkecil hingga terbesar. Jarak titik terdekat dimisalkan dengan $distMid$. Tahap ini memiliki kompleksitas $O(n \log n)$ karena perlu dilakukan sorting. Namun tahap ini dapat memiliki worst-case scenario $O(n^2)$ apabila seluruh hasil proyeksi titik memiliki jarak $\leq minDist$. Kompleksitas lower-bound ini akan lebih sering dijumpai seiring meningkatnya dimensi ruang vektor.

Setelah didapatkan pasangan titik terdekat pada strip tengah, maka $distMid$ dapat dibandingkan dengan $minDist$ untuk mendapatkan solusi global.

Berdasarkan deskripsi algoritma diatas, maka kompleksitas divide and conquer-nya adalah $O(n (\log n)^2)$ dengan worst-case scenario $O(n^2)$.

2. Source Code

Pada implementasi digunakan bahasa pemrograman pascal sebagai berikut :

1. Points.py

Pada program ini, koordinat titik memiliki range -1e5 sampai 1e5

```
import math
import random

class Point:
    LIMIT = 1e5
    eucDistCnt = 0
    def __init__(self, dim = 3, arr = None):
        # constuctor berupa init(dim) atau init(arr)
        self.__r = []
        if (arr is not None):
            self.__d = len(arr)
            for i in range(len(arr)):
                self.__r.append(arr[i])
        else:
            self.__d = dim
            for i in range(dim):
                self.__r.append(random.uniform(-Point.LIMIT,Point.LIMIT))

    def getDim(self):
        return self.__d

    def getXi(self,i):
        if (i >= self.__d):
            return 0
        else:
            return self.__r[i]

    def getFullCoor(self):
        return self.__r

    def eucDist(self,p):
        # menghitung jarak euclidean dengan titik p
        ans = 0
        for i in range(min(self.__d,p.__d)):
            ans += (self.__r[i]-p.__r[i])**2
        Point.eucDistCnt += 1
        return math.sqrt(ans)

    @staticmethod
    def resetEucDistCnt():
        # mereset perhitungan pemanggilan eucDist
        Point.eucDistCnt = 0

    def __str__(self):
```

```

        l = []
        for i in range(self.__d):
            l.append("{:.2f}".format(self.__r[i]))
        return '['+', '.join(l)+']'

    def __repr__(self):
        return str(self)

```

2. Sorter.py

Pada program ini, implementasi sorting menggunakan algoritma mergesort.

```

from Point import *

def sort(arrP,idxCMP):
    mergeSort(arrP,idxCMP,0,len(arrP)-1)

def mergeSort(arrP, idxCMP, l, r):
    if (l >= r):
        return
    mid = (l+r)//2
    mergeSort(arrP,idxCMP,l,mid)
    mergeSort(arrP,idxCMP,mid+1,r)
    b = []
    i = l
    j = mid+1
    while (i <= mid and j <= r):
        if (arrP[i].getXi(idxCMP) < arrP[j].getXi(idxCMP)):
            b.append(arrP[i])
            i += 1
        else:
            b.append(arrP[j])
            j += 1
    while (i <= mid):
        b.append(arrP[i])
        i += 1
    while (j <= r):
        b.append(arrP[j])
        j += 1
    for i in range(l,r+1):
        arrP[i] = b[i-l]
    return

```

3. SpaceOfPoints.py

```

import PointSorter
from Point import Point

class SpaceOfPoints:
    INF = 1e9

```

```

    def __init__(self, d = 3, n = 10, points = None, arrPoints = None, fileDir
= None):
        # konstruktor berupa init(d,n), init(points), init(arrPoints),
init(fileDir)
        if (fileDir is not None):
            with open(fileDir) as f:
                self.__n, self.__d = [int(x) for x in next(f).split()]
                self.__points = []
                for line in f:
                    self.__points.append(Point(arr = [int(x) for x in
line.split()])))
        elif (arrPoints is not None):
            self.__points = [Point(arr = list(arrPoints[i])) for i in
range(len(arrPoints))]
            self.__n = len(arrPoints)
            self.__d = d
        elif (points is not None):
            self.__points = points
            self.__n = len(points)
            self.__d = d
        else :
            self.__points = [None for i in range(n)]
            for i in range(n):
                self.__points[i] = Point(d)
            self.__n = n
            self.__d = d

    def isEmpty(self):
        return self.__n == 0

    def getPoints(self):
        return self.__points

    def getN(self):
        return self.__n

    def getD(self):
        return self.__d

    def getFilteredPointsByIdx(self, l, r):
        # mengembalikan himpunan titik yang dibagi berdasarkan indeks
        fPoints = self.__points[l:r+1]
        return SpaceOfPoints(len(fPoints), self.__d, fPoints)

    def getFilteredPointsByMinMax(self, minX, maxX, di):
        # mengembalikan himpunan titik yang dibagi berdasarkan nilai aksis ke-
di
        fPoints = []

```

```

        for i in range(self.__n):
            if minX <= self.__points[i].getXi(di) <= maxX:
                fPoints.append(self.__points[i])
        return SpaceOfPoints(self.__d, points = fPoints)

    def dncShortestPair(self):
        # set up solusi divide and conquer
        # melakukan presort sumbu pertama
        sortedPoint = self.__points.copy()
        PointSorter.sort(sortedPoint,0)
        return
SpaceOfPoints(self.__d,points=sortedPoint).__dncShortestPairRec()

    def __dncShortestPairRec(self):
        # solusi divide and conquer utama
        if (self.__n == 1) :
            return SpaceOfPoints.INF, None, None
        elif (self.__n == 2):
            return self.__points[0].eucDist(self.__points[1]),
self.__points[0], self.__points[1]
        else:
            mid = (self.__n-1)//2
            qL = self.getFilteredPointsByIdx(0,mid)
            qR = self.getFilteredPointsByIdx(mid+1,self.__n-1)
            dL, pLA, pLB = qL.__dncShortestPairRec()
            dR, pRA, pRB = qR.__dncShortestPairRec()
            if (dL < dR):
                pA = pLA
                pB = pLB
                minDist = dL
            else:
                pA = pRA
                pB = pRB
                minDist = dR
            xMid = self.__points[mid].getXi(0)
            qMid = self.getFilteredPointsByMinMax(xMid-minDist,xMid+minDist,0)
            dMid, pMidA, pMidB = qMid.__findClosestSparse(minDist)
            if (dMid < minDist):
                pA = pMidA
                pB = pMidB
                minDist = dMid
            return minDist, pA, pB

    def __findClosestSparse(self,minRange):
        # mencari jarak terdekat pada kasus titik yang jaraknya <= minRange
dari L
        if (self.isEmpty()) :
            return SpaceOfPoints.INF

```

```

        else:
            minDist = SpaceOfPoints.INF
            pA = None
            pB = None
            sortedPoints = self.__points.copy()
            PointSorter.sort(sortedPoints, self.__d-1)
            for i in range(self.__n):
                j = i+1
                now = sortedPoints[i]
                while (j < self.__n and sortedPoints[j].getXi(self.__d-1)-
now.getXi(self.__d-1) < minRange):
                    dist = now.eucDist(sortedPoints[j])
                    if (dist < minDist):
                        minDist = dist
                        pA = now
                        pB = sortedPoints[j]
                    j += 1
            return minDist, pA, pB

def bruteShortestPair(self):
    # solusi bruteforce
    minDist = SpaceOfPoints.INF
    pA = None
    pB = None
    for i in range(self.__n-1):
        for j in range(i+1, self.__n):
            dist = self.__points[i].eucDist(self.__points[j])
            if (dist < minDist):
                minDist = dist
                pA = self.__points[i]
                pB = self.__points[j]
    return minDist, pA, pB

def printToFile(self, fileDir):
    f = open(fileDir, 'w')
    print(self.__n, self.__d, file=f)
    for i in range(self.__n):
        for j in range(self.__d):
            print(self.__points[i].getXi(j), end=' ', file=f)
        print(file=f)

def __str__(self):
    return str(self.__points)

def __repr__(self):
    return str(self)

```


4. main.py

```
from SpaceOfPoints import *
from Point import *
import time
import matplotlib.pyplot as plt
import numpy as np
import platform

print("=====")
print("\\"Closest Pair Finder\\"")
print("Program untuk mencari pasangan titik terdekat dari sebuah himpunan titik")
print("Oleh : Muhammad Habibi Husni")
print("=====")
print("Pilihan metode membangkitkan titik")
print("1. Random")
print("2. Membaca file")

while (True):
    pil = str(input("Masukkan pilihan anda: "))
    if (pil == '1'):
        n = int(input("Masukkan banyak poin (n >= 2) : "))
        while (n < 2):
            print("Masukkan nilai yang valid!")
            n = int(input("Masukkan banyak poin (n) : "))
        d = int(input("Masukkan banyak dimensi (d >= 1) : "))
        while (d < 1):
            print("Masukkan nilai yang valid!")
            d = int(input("Masukkan banyak dimensi (d >= 1) : "))
        A = SpaceOfPoints(d,n)
        break
    elif (pil == '2'):
        print("Format file input adalah sebagai berikut : ")
        print("n d")
        print("x0,0 x0,1 x0,2 ...")
        print("x1,0 x1,1 x1,2 ...")
        print("dengan n banyak poin, d banyak dimensi, dan xi,j koordinat dari axis ke-j dari titik ke-i")
        fileDir = str(input("Masukkan lokasi file (absolute) : "))
        A = SpaceOfPoints(fileDir=fileDir)
        break
    print("Masukan tidak valid!")
print()

print("Processor name : ", platform.processor())
print("Solusi bruteforce : ")
st = time.time()
dist, pA, pB = A.bruteShortestPair()
```

```

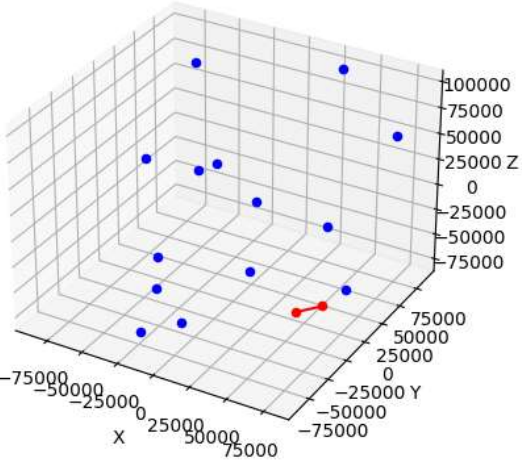
end = time.time()
print("jarak terdekat = ", dist)
print("Titik 1 = ", pA)
print("Titik 2 = ", pB)
print("Banyak eucDist dipanggil = ", Point.eucDistCnt)
print("Runtime = ", end-st)
print()

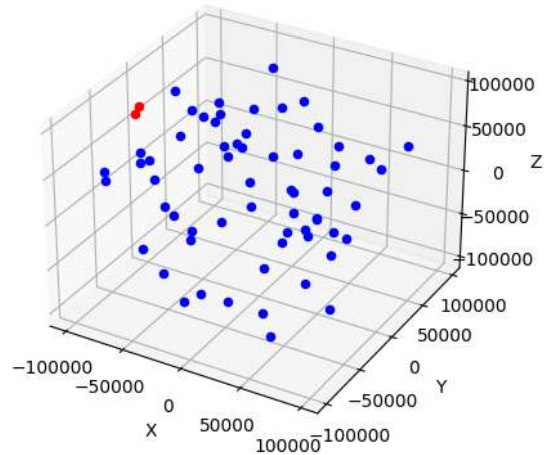
Point.resetEucDistCnt()
print("Solusi divide and conquer : ")
st = time.time()
dist, pA, pB = A.dncShortestPair()
end = time.time()
print("Jarak terdekat = ", dist)
print("Titik 1 = ", pA)
print("Titik 2 = ", pB)
print("Banyak eucDist dipanggil = ", Point.eucDistCnt)
print("Runtime = ", end-st)
print()

if (A.getD() <= 3):
    pil = input("Apakah Anda ingin melihat visualisasi titik? (y/n) : ")
    if (pil == 'y'):
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')
        for i in range(A.getN()):
            p = A.getPoints()[i]
            if (p != pA and p != pB):
                col = 'blue'
            else:
                col = 'red'
            ax.scatter(p.getXi(0),p.getXi(1),p.getXi(2),c=col)
        x = np.linspace(pA.getXi(0),pB.getXi(0))
        y = np.linspace(pA.getXi(1),pB.getXi(1))
        z = np.linspace(pA.getXi(2),pB.getXi(2))
        plt.plot(x,y,z,c='red')
        plt.show()
    else:
        print("Dimensi >= 3, tidak ada visualisasi")
print("Program selesai...")

```

4. Input dan Output

Kasus	Output
<p>n = 16 d = 3</p>	<pre> Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 16 Masukkan banyak dimensi (d >= 1) : 3 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : jarak terdekat = 23533.34951096576 Titik 1 = [74544.51,-24540.96,-41983.19] Titik 2 = [56087.84,-23845.41,-56566.94] Banyak eucDist dipanggil = 120 Runtime = 0.0 Solusi divide and conquer : Jarak terdekat = 23533.34951096576 Titik 1 = [74544.51,-24540.96,-41983.19] Titik 2 = [56087.84,-23845.41,-56566.94] Banyak eucDist dipanggil = 83 Runtime = 0.0009996891021728516 </pre> 
<p>n = 64 d = 3</p>	<pre> Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 64 Masukkan banyak dimensi (d >= 1) : 3 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : jarak terdekat = 12111.788649765722 Titik 1 = [-79968.79,-37902.59,95911.47] Titik 2 = [-84414.00,-36486.17,84734.30] Banyak eucDist dipanggil = 2016 Runtime = 0.003002166748046875 Solusi divide and conquer : Jarak terdekat = 12111.788649765722 Titik 1 = [-79968.79,-37902.59,95911.47] Titik 2 = [-84414.00,-36486.17,84734.30] Banyak eucDist dipanggil = 742 Runtime = 0.0019989013671875 </pre>

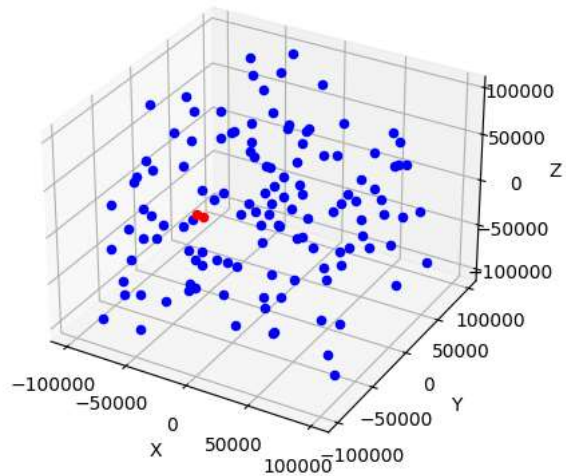


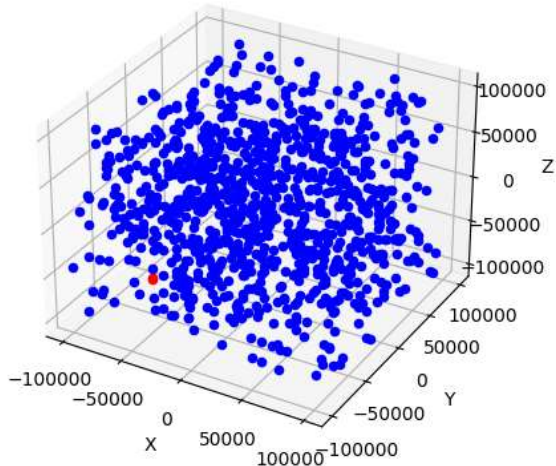
n = 128
d = 3

```
Pilihan metode membangkitkan titik
1. Random
2. Membaca file
Masukkan pilihan anda: 1
Masukkan banyak poin (n >= 2) : 128
Masukkan banyak dimensi (d >= 1) : 3

Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD
Solusi bruteforce :
jarak terdekat = 8236.054314199277
Titik 1 = [1604.05,-89871.85,52069.50]
Titik 2 = [-196.34,-95240.95,58049.79]
Banyak eucDist dipanggil = 8128
Runtime = 0.011134862899780273

Solusi divide and conquer :
Jarak terdekat = 8236.054314199277
Titik 1 = [1604.05,-89871.85,52069.50]
Titik 2 = [-196.34,-95240.95,58049.79]
Banyak eucDist dipanggil = 2610
Runtime = 0.0059986114501953125
```



<p>n = 1000 d = 3</p>	<pre>Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 1000 Masukkan banyak dimensi (d >= 1) : 3 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : jarak terdekat = 747.6892362453609 Titik 1 = [-71813.17,-36200.68,-87277.91] Titik 2 = [-72379.73,-35761.39,-87065.60] Banyak eucDist dipanggil = 499500 Runtime = 0.5588326454162598 Solusi divide and conquer : Jarak terdekat = 747.6892362453609 Titik 1 = [-71813.17,-36200.68,-87277.91] Titik 2 = [-72379.73,-35761.39,-87065.60] Banyak eucDist dipanggil = 44808 Runtime = 0.07609033584594727</pre> 
<p>n = 64 d = 4</p>	<pre>Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 64 Masukkan banyak dimensi (d >= 1) : 4 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : jarak terdekat = 11688.476842681803 Titik 1 = [32208.52,21622.05,-77433.83,-33465.86] Titik 2 = [26712.75,24512.26,-80529.35,-42872.32] Banyak eucDist dipanggil = 2016 Runtime = 0.0029916763305664062 Solusi divide and conquer : Jarak terdekat = 11688.476842681803 Titik 1 = [32208.52,21622.05,-77433.83,-33465.86] Titik 2 = [26712.75,24512.26,-80529.35,-42872.32] Banyak eucDist dipanggil = 928 Runtime = 0.0019979476928710938</pre>
<p>n = 64 d = 10</p>	<pre>Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 64 Masukkan banyak dimensi (d >= 1) : 10 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : jarak terdekat = 87974.356817559 Titik 1 = [33847.46,64204.02,-92154.70,36601.19,65807.15,-51566.26,43765.07,-49161.10,-75070.71,61511.81] Titik 2 = [58368.64,89828.91,-63014.05,73742.36,25998.58,-21686.87,73496.19,-32610.68,-68381.77,37578.11] Banyak eucDist dipanggil = 2016 Runtime = 0.004998445510864258 Solusi divide and conquer : Jarak terdekat = 87974.356817559 Titik 1 = [58368.64,89828.91,-63014.05,73742.36,25998.58,-21686.87,73496.19,-32610.68,-68381.77,37578.11] Titik 2 = [33847.46,64204.02,-92154.70,36601.19,65807.15,-51566.26,43765.07,-49161.10,-75070.71,61511.81] Banyak eucDist dipanggil = 2983 Runtime = 0.009020328521728516</pre>

n = 1000 d = 10	Pilihan metode membangkitkan titik 1. Random 2. Membaca file Masukkan pilihan anda: 1 Masukkan banyak poin (n >= 2) : 1000 Masukkan banyak dimensi (d >= 1) : 10 Processor name : AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD Solusi bruteforce : Jarak terdekat = 39089.407040704464 Titik 1 = [66650.15,1359.04,65718.94,-73758.53,56390.78,90409.77,39520.01,33217.04,-57567.21,-40562.18] Titik 2 = [85026.75,-2760.70,68543.16,-49268.98,57850.12,83933.33,30731.10,50745.30,-46824.38,-45212.82] Banyak eucDist dipanggil = 499500 Runtime = 1.0815155506134033 Solusi divide and conquer : Jarak terdekat = 39089.407040704464 Titik 1 = [85026.75,-2760.70,68543.16,-49268.98,57850.12,83933.33,30731.10,50745.30,-46824.38,-45212.82] Titik 2 = [66650.15,1359.04,65718.94,-73758.53,56390.78,90409.77,39520.01,33217.04,-57567.21,-40562.18] Banyak eucDist dipanggil = 478524 Runtime = 1.1916723251342773
--------------------	---

5. Pranala Github

https://github.com/habibibi/Tucil2_13521169

6. Check List Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada 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	✓	