TUGAS KECIL 02

MENCARI PASANGAN TITIK TERDEKAT DENGAN ALGORITMA DIVIDE AND CONQUER
IF2211 – STRATEGI ALGORITMA

Disusun oleh:

Muhammad Fadhil Amri	13521066
Nathan Tenka	13521172

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

Daftar Isi

Bab I : Algoritma Divide and Conquer Pencarian Titik Terdekat

1.1 *Overview* Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* adalah algoritma yang memecah persoalan menjadi beberapa upa-persoalan, menyelesaikan tiap upa-persoalan secara langsung jika sudah cukup kecil, dan menggabungkan hasil dari tiap upa-persoalan sehingga didapat solusi dari persoalan awal. Tiap upa-persoalan memiliki karakteristik yang sama dengan persoalan awal. Berikut adalah skema umum algoritma divide and conquer diambil dari slide perkuliahan IF2211 tahun 2021 oleh Bapak Rinaldi Munir :

Skema Umum Algoritma *Divide and Conquer*

```
procedure DIVIDEandCONQUER(input P : problem, n : integer)
{ Menyelesaikan persoalan P dengan algoritma divide and conquer
  Masukan: masukan persoalan P berukuran n
  Luaran: solusi dari persoalan semula  }
Deklarasi
   r : integer

Algoritma
  if n ≤ n0 then  {ukuran persoalan P sudah cukup kecil }
     SOLVE  persoalan P yang berukuran n ini
  else
     DIVIDE  menjadi r upa-persoalan, P1, P2, ..., Pr, yang masing-masing berukuran n1, n2, ..., nr
     for  masing-masing P1, P2, ..., Pr, do
          DIVIDEandCONQUER(Pi, ni)
     endfor
     COMBINE  solusi dari P1, P2, ..., Pr  menjadi solusi persoalan semula
  endif
```

Kompleksitas algoritma *divide and conquer*: $T(n) = \begin{cases} g(n) & , n \leq n_0 \\ T(n_1) + T(n_2) \ldots + T(n_r) + f(n) & , n > n_0 \end{cases}$

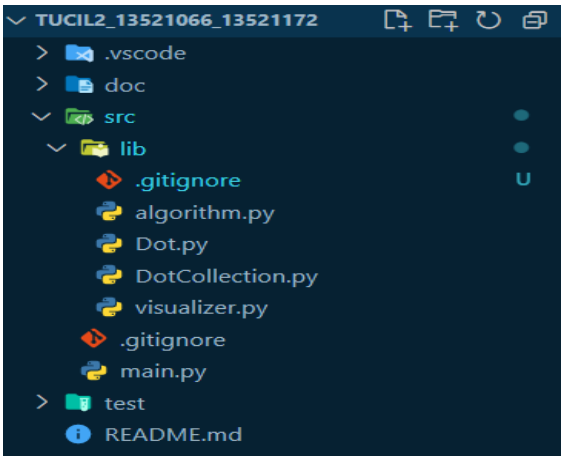
1.2 Penerapan Algoritma *Divide and Conquer*

Berikut adalah algoritma pencarian titik terdekat.

1. Terima banyak titik, banyak dimensi, dan masing-masing titik.
2. Urutkan tiap titik berdasarkan posisinya di salah satu sumbu. Dalam program ini kami mengurutkan titik berdasarkan posisinya pada sumbu pertama (sumbu X).
3. Bila jumlah titik hanya dua atau tiga, langsung hitung jarak antar masing-masing titik dan kembalikan jarak terkecilnya.
4. Jika jumlah titik lebih banyak, bagi titik-titik ke dalam dua daerah dan hitung jarak terkecil di masing-masing daerah. Ambil jarak yang lebih kecil (misalkan d) sebagai batas untuk perhitungan jarak titik antar daerah.
5. Ambil titik-titik yang jaraknya dengan garis pemisah (misal ada di posisi titik tengah) lebih kecil atau sama dengan d.
6. Untuk tiap titik, hitung jaraknya dengan titik lain yang selisih posisi di tiap sumbunya <= d.
7. Jika jaraknya lebih kecil dari d, simpan sebagai nilai d yang baru.

Bab II: Kode Program

Program ditulis menggunakan bahasa pemrograman Python versi 3.9.5 dan 3.10.6 pada sistem operasi Windows 11. Program terdiri atas 1 file **main.py** dan 1 folder **lib** yang terdiri atas 4 buah file yang akan digunakan pada *main program*. Berikut posisi file pada folder.



2.1 main.py

File ini berisi main program yang terdiri atas input, pemrosesan, dan output.

```python
from lib import DotCollection
from lib import algorithm
from lib import visualizer


if __name__ == "__main__":
    ### Input ###
    valid = False
    while (not (valid)):
        n = int(input("Masukkan banyak titik (n): "))
        nDim = int(input("Masukkan dimensi titik (nDim): "))
        if (n <= 1 or nDim < 1):
            print("Masukan tidak valid. Pastikan banyak titik > 1 dan dimensi titik >= 1")
        else:
            valid = True

    ### Process ###

    ## Generating Dots ##
    listOfDotDnC = DotCollection.DotCollection(n, nDim)
    # algorithm.sortArrOfDot(listOfDotDnC.getArrOfDot())
    listOfDotBF = DotCollection.DotCollection()
    listOfDotDnC.copy(listOfDotBF)
    ## Calculate Shortest Distance, time, etc ##
    # Divide and Conquer
    algorithm.divideAndConquerShortestDistance(listOfDotDnC)
    # Brute Force
    algorithm.bruteForceShortestDistance(listOfDotBF)

    ### Output ###

    choice = input("Simpan output ke file ? Y/N\n")
    if (choice == 'Y'):
        fileName = input("Masukkan nama file : ")
        f = open(fileName, "w", encoding="utf-8")
        f.write("Titik-titik :\n")
        listOfDotBF.printArrToFile(f)
```

```
        f.write("\n")
        # Divide and Conquer
        f.write("Hasil :\n")
        f.write("Divide and Conquer\n")
        f.write("Closest Points: (" + str(listOfDotDnC.getClosestPoints()
                                          [0].getCoordinate()) + "," + str(listOfDotDnC.getClosestPoints()[1].getCoordinate()) + ")\n")
        f.write("Distance: " + str(listOfDotDnC.getShortestDistance()) + "\n")

        # Brute Force
        f.write("Brute Force\n")
        f.write("Closest Points: (" + str(listOfDotBF.getClosestPoints()
                                          [0].getCoordinate()) + "," + str(listOfDotBF.getClosestPoints()[1].getCoordinate()) + ")\n")
        f.write("Distance: " + str(listOfDotBF.getShortestDistance()) + "\n")

        ## Banyak Operasi Perhitungan Rumus Euclidean ##
        # Divide and Conquer
        f.write("N perhitungan\n")
        f.write("Divide and Conquer: " + str(listOfDotDnC.getNStep()) + "\n")

        # Brute Force
        f.write("Brute Force: " + str(listOfDotBF.getNStep()) + "\n")

        ## Execution Time (Spesifikasikan komputer yang digunakan) ##
        # Divide and Conquer
        f.write("Execution Time\n")
        f.write("Divide and Conquer: " +
                str(listOfDotDnC.getSolvingTime()) + "s\n")

        # Brute Force
        f.write("Brute Force: " + str(listOfDotBF.getSolvingTime()) + "s\n")
        f.close()

    ## Sepasang Titik Terdekat dan Jaraknya ##
```

```
# Divide and Conquer
print("Divide and Conquer")
print("Closest Points: ", "(", listOfDotDnC.getClosestPoints()
      [0].getCoordinate(), ",", listOfDotDnC.getClosestPoints()[1].getCoordinate(), ")")
print("Distance: ", listOfDotDnC.getShortestDistance())

# Brute Force
print("Brute Force")
print("Closest Points: ", "(", listOfDotBF.getClosestPoints()
      [0].getCoordinate(), ",", listOfDotBF.getClosestPoints()[1].getCoordinate(), ")")
print("Distance: ", listOfDotBF.getShortestDistance())

## Banyak Operasi Perhitungan Rumus Euclidean ##
# Divide and Conquer
print("N perhitungan")
print("Divide and Conquer: ", listOfDotDnC.getNStep())

# Brute Force
print("Brute Force: ", listOfDotBF.getNStep())

## Execution Time (Spesifikasikan komputer yang digunakan) ##
# Divide and Conquer
print("Execution Time")
print("Divide and Conquer: ", listOfDotDnC.getSolvingTime(), "s")

# Brute Force
print("Brute Force: ", listOfDotBF.getSolvingTime(), "s")

# Komputer yang digunakan
print("Computer", end=" ")
print(model)

## Visualisasi titik ##
if (nDim == 3):  # 3D
    visualizer.show3D(listOfDotBF)
elif (nDim == 2):  # 2D
    visualizer.show2D(listOfDotBF)
elif (nDim == 1):  # 1D
    visualizer.show1D(listOfDotBF)
else:
    print("Kumpulan titik tidak dapat divisualisasikan")
```

2.2 Dot.py

File ini berisi kelas Dot yang merepresentasikan sebuah titik beserta atribut dan fungsionalitas yang dimilikinya.

```python
from random import uniform


...
class Dot:
    # Constructor
    def __init__(self, nDim):
        # Attributes
        self._nDim = nDim
        self._color = 'blue'
        self._coordinate = list()  # Pembangkitan coordinate acak
        for i in range(nDim):
            # Ini buat starting, mungkin nanti direvisi lagi
            # tipe coordinate adalah floating point
            self._coordinate.append(uniform(-10000, 10000))

    # Getter
    def getNDim(self):
        return self._nDim

    def getColor(self):
        return self._color

    def getCoordinate(self):
        return self._coordinate

    # Setter
    def setColor(self, color):
        self._color = color
```

2.3 DotCollection.py

File ini berisi kelas DotCollection yang merepresentasikan kumpulan titik yang akan dicari pasangan titik terdekatnya.

```python
from . import Dot


...
class DotCollection:
    # Insert dot
    def __init__(self, *args):  # default (), user-defined (nDots, dim)
        self.arrOfDot = []
        self.nDots = 0
        self.solvingTime = 0
        self.nStep = 0
        self.closestIndexes = [0, 0]
        self.closestPoints = []
        self.shortest_distance = 0
        if (len(args) == 2):
            self.nDots = args[0]
            for i in range(self.nDots):
                self.arrOfDot.append(Dot.Dot(args[1]))

    # mengcopy nilai
    def copy(self, listOfDot):
        listOfDot.nDots = self.nDots
        for i in self.arrOfDot:
            listOfDot.arrOfDot.append(i)

    def addDot(self, d):
        self.arrOfDot.append(d)
        self.nDots += 1

    # Setter
    def setClosestIndexes(self, idx1, idx2):
        self.closestIndexes[0] = idx1
        self.closestIndexes[1] = idx2

    def setNStep(self, steps):
        self.nStep = steps
```

```
    def setSolvingTime(self, time):
        self.solvingTime = time

    def setShortestDistance(self, shortest_distance):
        self.shortest_distance = shortest_distance

    # Getter
    def getClosestIndexes(self):
        return self.closestIndexes

    def getNStep(self):
        return self.nStep

    def getSolvingTime(self):
        return self.solvingTime

    def getArrOfDot(self):
        return self.arrOfDot

    def getNDots(self) :
        return self.nDots

    def getShortestDistance(self):
        return self.shortest_distance

    def getClosestPoints(self):
        return (self.arrOfDot[self.closestIndexes[0]], self.arrOfDot[self.closestIndexes[1]])
```

2.3 algorithm.py

File ini berisi fungsi dan prosedur yang digunakan untuk algoritma pencarian titik terdekat, baik secara *divide and conquer* ataupun *brute force*.

```
import math
import time



def swap(arrOfDot, a, b):
    temp = arrOfDot[a]
    arrOfDot[a] = arrOfDot[b]
    arrOfDot[b] = temp


# Sort arrOfDot berdasarkan coordinate[0] (sumbu X), memakai algoritma quicksort dengan pemilihan pivot elemen tengah
def partition(arrOfDot, i, j, axis):
    # Ambil pivot dari elemen acak
    pivotIndex = (i+j)//2
    pivot = arrOfDot[pivotIndex].getCoordinate()[axis]
    p = i
    q = j
    while True:
        while arrOfDot[p].getCoordinate()[axis] < pivot:
            p += 1
        while arrOfDot[q].getCoordinate()[axis] > pivot:
            q -= 1
        if (p < q) :
            swap(arrOfDot, p, q)
            p += 1
            q -= 1
        else :
            break
    return q
```

```
def sortArrOfDot(arrOfDot, i=0, j=-1, axis=0):
    if (j == -1):
        j = len(arrOfDot)-1
    if (i < j):
        pos = partition(arrOfDot, i, j, axis)
        sortArrOfDot(arrOfDot, i, pos, axis)
        sortArrOfDot(arrOfDot, pos+1, j, axis)

# Fungsi Untuk menghitung jarak antara dua buah titik


def calculateDistance(dot1, dot2):
    distance = 0  # inisialisasi
    # Penghitungan d = sqrt((x1-x2)^2 + (y1-y2)^2 + ...)
    for i in range(dot1.getNDim()):
        distance += math.pow((dot1.getCoordinate()[i] -
                              dot2.getCoordinate()[i]), 2)
    distance = math.sqrt(distance)
    return distance

# Fungsi untuk mencari shortest distance dengan algoritma brute force


def bruteForceShortestDistance(listOfDot):
    arrOfDot = listOfDot.getArrOfDot()
    shortest_distance = 0
    closest_indexes = [-1, -1]
    num_step = 0
    start_time = time.time()

    if (len(arrOfDot) > 1):
        shortest_distance = calculateDistance(arrOfDot[0], arrOfDot[1])
        closest_indexes = [1, 0]
```

```
        for i in range(len(arrOfDot)):
            for j in range(i+1, len(arrOfDot)):
                num_step += 1
                distance = calculateDistance(arrOfDot[i], arrOfDot[j])
                if (distance < shortest_distance):
                    shortest_distance = distance
                    closest_indexes[0] = i
                    closest_indexes[1] = j
        listOfDot.getArrOfDot()[closest_indexes[0]].setColor("red")
        listOfDot.getArrOfDot()[closest_indexes[1]].setColor("red")
    exec_time = time.time() - start_time
    listOfDot.setShortestDistance(shortest_distance)
    listOfDot.setClosestIndexes(closest_indexes[0], closest_indexes[1])
    listOfDot.setNStep(num_step)
    listOfDot.setSolvingTime(exec_time)


def searchShortestPartition(arrOfDot, i, j, numStep=0):
    # Diasumsikan i adalah indeks awal partisi dan j indeks akhir partisi
    if (j-i == 1):  # Hanya ada 1 pasang titik
        # kembalikan tuple berisi shortest distance dan closest index
        return calculateDistance(arrOfDot[i], arrOfDot[j]), [i, j], numStep+1
    elif (j-i == 2):  # Ada 3 titik (penanganan kasus ganjil)
        dist1 = calculateDistance(arrOfDot[i], arrOfDot[i+1])
        dist2 = calculateDistance(arrOfDot[i], arrOfDot[j])
        dist3 = calculateDistance(arrOfDot[i+1], arrOfDot[j])
        if (dist1 <= dist2):
            if (dist1 <= dist3):
                return dist1, [i, i+1], numStep+3
            else:
                return dist3, [i+1, j], numStep+3
        else:
            if (dist2 <= dist3):
                return dist2, [i, j], numStep+3
            else:
                return dist3, [i+1, j], numStep+3
```

```
    else:  # Ada lebih dari 3 titik
        mid = (i+j)//2
        dist1, closestIndex1, numStep = searchShortestPartition(
            arrOfDot, i, mid, numStep)
        dist2, closestIndex2, numStep = searchShortestPartition(
            arrOfDot, mid+1, j, numStep)
        if (dist1 <= dist2):
            minDist = dist1
            closestIndex = closestIndex1
        else:
            minDist = dist2
            closestIndex = closestIndex2
        # Cari titik-titik yang dipisah garis dengan selisih jarak pada absis maksimal d
        midX = arrOfDot[mid].getCoordinate()[0]
        nDim = arrOfDot[0].getNDim()
        p = mid
        while (p >= i and abs(arrOfDot[p].getCoordinate()[0]-midX) <= minDist):
            q = mid+1
            # Sementara masih cek semua titik di dalam strip
            while (q <= j and abs(arrOfDot[q].getCoordinate()[0]-midX) <= minDist):
                inRange = True
                for dim in range(1, nDim):
                    # Cek apakah ada jarak pada sumbu tertentu yang lebih besar dari minDist
                    if (abs(arrOfDot[q].getCoordinate()[dim]-arrOfDot[p].getCoordinate()[dim]) > minDist):
                        inRange = False
                        break
                if (inRange):
                    dist3 = calculateDistance(arrOfDot[p], arrOfDot[q])
                    numStep += 1
                    if (dist3 < minDist):
                        minDist = dist3
                        closestIndex = [p, q]
                q += 1
            p -= 1
        return minDist, closestIndex, numStep
```

```
def divideAndConquerShortestDistance(listOfDot):
    arrOfDot = listOfDot.getArrOfDot()
    shortest_distance = 0
    closest_indexes = [0, 0]
    num_step = 0

    if (len(arrOfDot) > 1):
        sortArrOfDot(arrOfDot)
        start_time = time.time()
        shortest_distance, closest_indexes, num_step = searchShortestPartition(
            arrOfDot, 0, listOfDot.getNDots()-1)
        listOfDot.getArrOfDot()[closest_indexes[0]].setColor("red")
        listOfDot.getArrOfDot()[closest_indexes[1]].setColor("red")
    else:
        start_time = time.time()
        shortest_distance = 0
    exec_time = time.time() - start_time
    listOfDot.setShortestDistance(shortest_distance)
    listOfDot.setClosestIndexes(closest_indexes[0], closest_indexes[1])
    listOfDot.setNStep(num_step)
    listOfDot.setSolvingTime(exec_time)
```

2.4 visualizer.py

File ini berisi fungsi dan prosedur yang digunakan untuk melakukan visualisasi kumpulan titik dan pasangan titik terdekatnya.

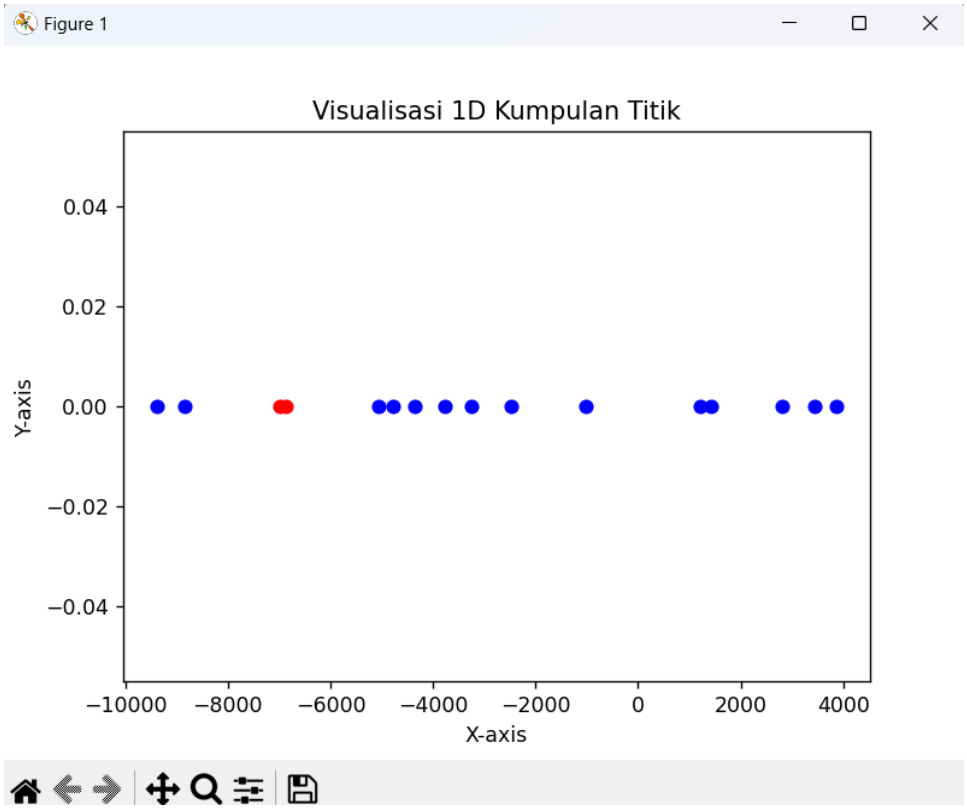
```python
import matplotlib.pyplot as plt


def show3D(listOfDot):
    # Fungsi untuk menampilkan visualisasi 3 dimensi dari titik-titik
    fig = plt.figure()

    ax = fig.add_subplot(projection='3d')
    plt.title("Visualisasi 3D Kumpulan Titik")
    ax.set_xlabel('X-axis')
    ax.set_ylabel('Y-axis')
    ax.set_zlabel('Z-axis')
    arrOfDot = listOfDot.getArrOfDot()
    dot1 = arrOfDot[listOfDot.getClosestIndexes()[0]]
    dot2 = arrOfDot[listOfDot.getClosestIndexes()[1]]
    for i in range(listOfDot.getNDots()):
        if ((i != listOfDot.getClosestIndexes()[0]) and (i != listOfDot.getClosestIndexes()[1])):
            coordinate = arrOfDot[i].getCoordinate()
            ax.scatter(coordinate[0], coordinate[1],
                       coordinate[2], color=arrOfDot[i].getColor())
    coordinate1 = arrOfDot[listOfDot.getClosestIndexes()[0]].getCoordinate()
    coordinate2 = arrOfDot[listOfDot.getClosestIndexes()[1]].getCoordinate()
    ax.scatter(coordinate1[0], coordinate1[1],
               coordinate1[2], color=arrOfDot[listOfDot.getClosestIndexes()[0]].getColor())
    ax.scatter(coordinate2[0], coordinate2[1],
               coordinate2[2], color=arrOfDot[listOfDot.getClosestIndexes()[1]].getColor())
    plt.show()


def show2D(listOfDot):
    # Fungsi untuk menampilkan visualisasi 2 dimensi dari titik-titik
    plt.title('Visualisasi 2D Kumpulan Titik')  # Title of the plot
    plt.xlabel('X-axis')  # X-Label
    plt.ylabel('Y-axis')  # Y-Label
    arrOfDot = listOfDot.getArrOfDot()
    for i in range(listOfDot.getNDots()):
        if ((i != listOfDot.getClosestIndexes()[0]) and (i != listOfDot.getClosestIndexes()[1])):
            coordinate = arrOfDot[i].getCoordinate()
            plt.scatter(coordinate[0], coordinate[1],
                        color=arrOfDot[i].getColor())

    coordinate1 = arrOfDot[listOfDot.getClosestIndexes()[
        0]].getCoordinate()
    coordinate2 = arrOfDot[listOfDot.getClosestIndexes()[1]].getCoordinate()
    plt.scatter(coordinate1[0], coordinate1[1],
                color=arrOfDot[listOfDot.getClosestIndexes()[0]].getColor())
    plt.scatter(coordinate2[0], coordinate2[1],
                color=arrOfDot[listOfDot.getClosestIndexes()[1]].getColor())
    plt.show()


def show1D(listOfDot):
    # Fungsi untuk menampilkan visualisasi 1 dimensi dari titik-titik
    plt.title('Visualisasi 1D Kumpulan Titik')  # Title of the plot
    plt.xlabel('X-axis')  # X-Label
    plt.ylabel('Y-axis')  # Y-Label
    arrOfDot = listOfDot.getArrOfDot()
    for i in range(listOfDot.getNDots()):
        if ((i != listOfDot.getClosestIndexes()[0]) and (i != listOfDot.getClosestIndexes()[1])):
            coordinate = arrOfDot[i].getCoordinate()
            plt.scatter(coordinate[0], 0,
                        color=arrOfDot[i].getColor())

    coordinate1 = arrOfDot[listOfDot.getClosestIndexes()[
        0]].getCoordinate()
    coordinate2 = arrOfDot[listOfDot.getClosestIndexes()[1]].getCoordinate()
    plt.scatter(coordinate1[0], 0,
                color=arrOfDot[listOfDot.getClosestIndexes()[0]].getColor())
    plt.scatter(coordinate2[0], 0,
                color=arrOfDot[listOfDot.getClosestIndexes()[1]].getColor())
    plt.show()
```

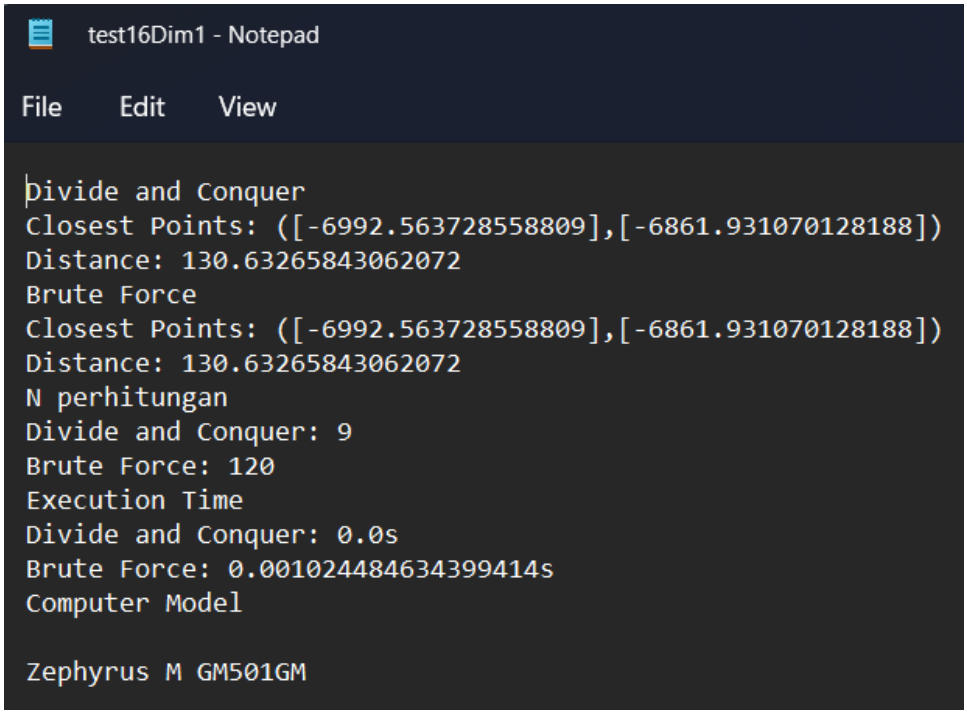
Bab III: Contoh Masukan dan Keluaran

3.1 n = 16

Dimensi 1

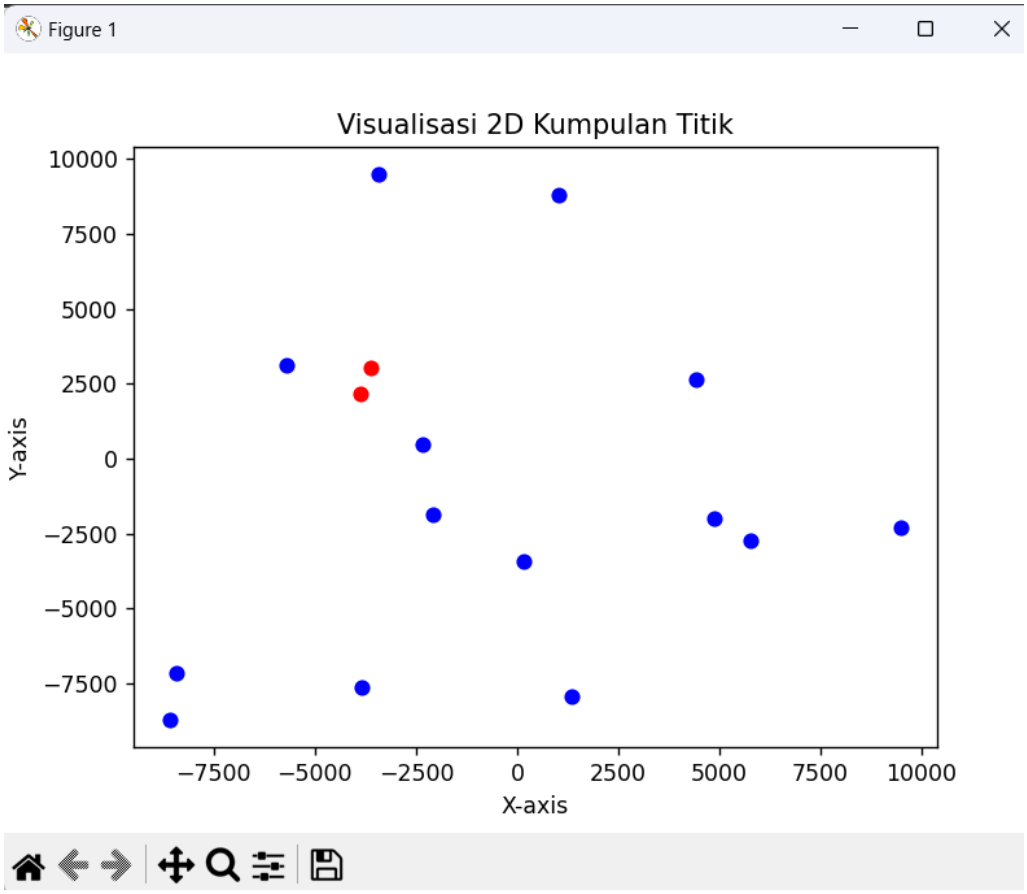
```
Masukkan banyak titik (n): 16
Masukkan dimensi titik (nDim): 1
Simpan output ke file ? Y/N
Y
Masukkan nama file : test16Dim1.txt
Divide and Conquer
Closest Points:  ( [-6992.563728558809] , [-6861.931070128188] )
Distance:  130.63265843062072
Brute Force
Closest Points:  ( [-6992.563728558809] , [-6861.931070128188] )
Distance:  130.63265843062072
N perhitungan
Divide and Conquer:  9
Brute Force:  120
Execution Time
Divide and Conquer:  0.0 s
Brute Force:  0.0010244846343994 14 s
Computer Model
Zephyrus M GM501GM
```





Dimensi 2

```
Masukkan banyak titik (n): 16
Masukkan dimensi titik (nDim): 2
Simpan output ke file ? Y/N
Y
Masukkan nama file : test16Dim2.txt
Divide and Conquer
Closest Points:  ( [-3881.3591128084354, 2166.07092525595] , [-3626.6386819592353, 3037.640808957138] )
Distance:  908.0289423068541
Brute Force
Closest Points:  ( [-3626.6386819592353, 3037.640808957138] , [-3881.3591128084354, 2166.07092525595] )
Distance:  908.0289423068541
N perhitungan
Divide and Conquer:  12
Brute Force:  120
Execution Time
Divide and Conquer:  0.0 s
Brute Force:  0.0010056495666503906 s
Computer Model
Zephyrus M GM501GM
```



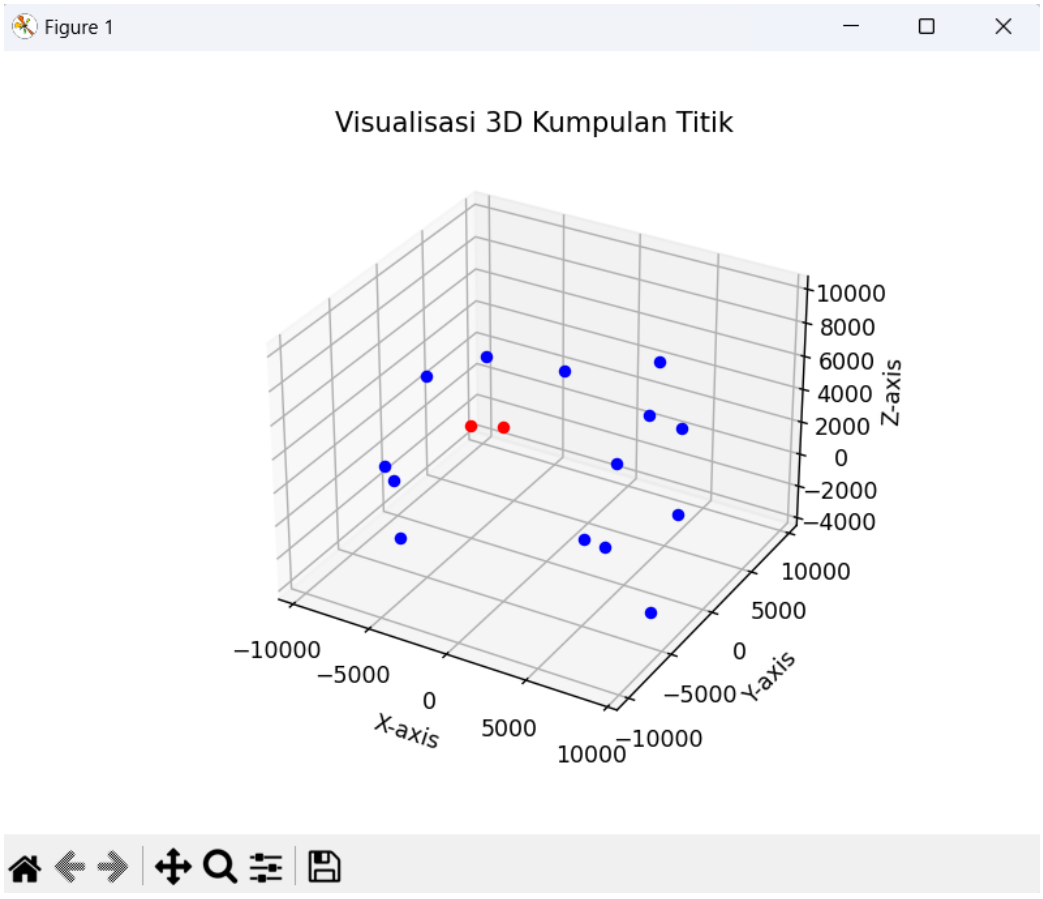
test16Dim2 - Notepad

File Edit View

```
Divide and Conquer
Closest Points: ([-3881.3591128084354, 2166.07092525595],[-3626.6386819592353, 3037.640808957138])
Distance: 908.0289423068541
Brute Force
Closest Points: ([-3626.6386819592353, 3037.640808957138],[-3881.3591128084354, 2166.07092525595])
Distance: 908.0289423068541
N perhitungan
Divide and Conquer: 12
Brute Force: 120
Execution Time
Divide and Conquer: 0.0s
Brute Force: 0.0010056495666503906s
Computer Model

Zephyrus M GM501GM
```

Dimensi 3

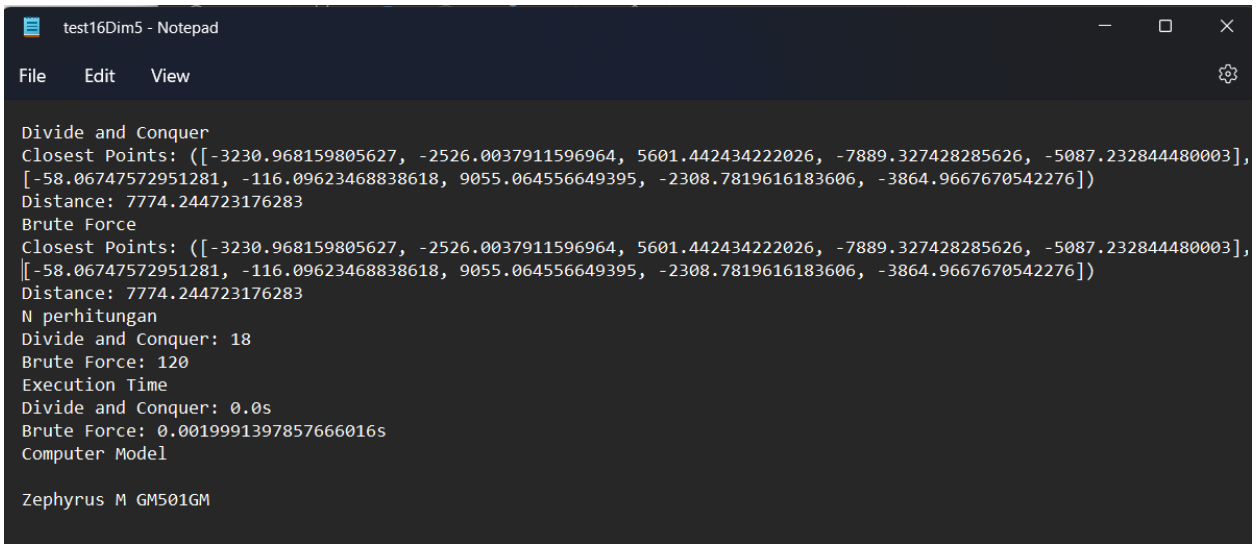
```
Masukkan banyak titik (n): 16
Masukkan dimensi titik (nDim): 3
Simpan output ke file ? Y/N
Y
Masukkan nama file : test16Dim3.txt
Divide and Conquer
Closest Points:  ( [-2377.7781558078996, -4384.522556434338, 5216.3856009325755] , [-455.150
4016656763, -4101.086069752105, 5524.135595723403] )
Distance:  1967.6238924841534
Brute Force
Closest Points:  ( [-455.1504016656763, -4101.086069752105, 5524.135595723403] , [-2377.7781
558078996, -4384.522556434338, 5216.3856009325755] )
Distance:  1967.6238924841534
N perhitungan
Divide and Conquer:  12
Brute Force:  120
Execution Time
Divide and Conquer:  0.0009946823120117188 s
Brute Force:  0.0010385513305664062 s
Computer Model
Zephyrus M GM501GM
```



Dimensi 5

```
Masukkan banyak titik (n): 16
Masukkan dimensi titik (nDim): 5
Simpan output ke file ? Y/N
Y
Masukkan nama file : test16Dim5.txt
Divide and Conquer
Closest Points:  ( [-3230.968159805627, -2526.0037911596964, 5601.442434222026, -7889.327428285626, -5087.232844480
003] , [-58.06747572951281, -116.09623468838618, 9055.064556649395, -2308.7819616183606, -3864.9667670542276] )
Distance:  7774.244723176283
Brute Force
Closest Points:  ( [-3230.968159805627, -2526.0037911596964, 5601.442434222026, -7889.327428285626, -5087.232844480
003] , [-58.06747572951281, -116.09623468838618, 9055.064556649395, -2308.7819616183606, -3864.9667670542276] )
Distance:  7774.244723176283
N perhitungan
Divide and Conquer:  18
Brute Force:  120
```

```
Execution Time
Divide and Conquer:  0.0 s
Brute Force:  0.0019991397857666016 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```



Dimensi 10

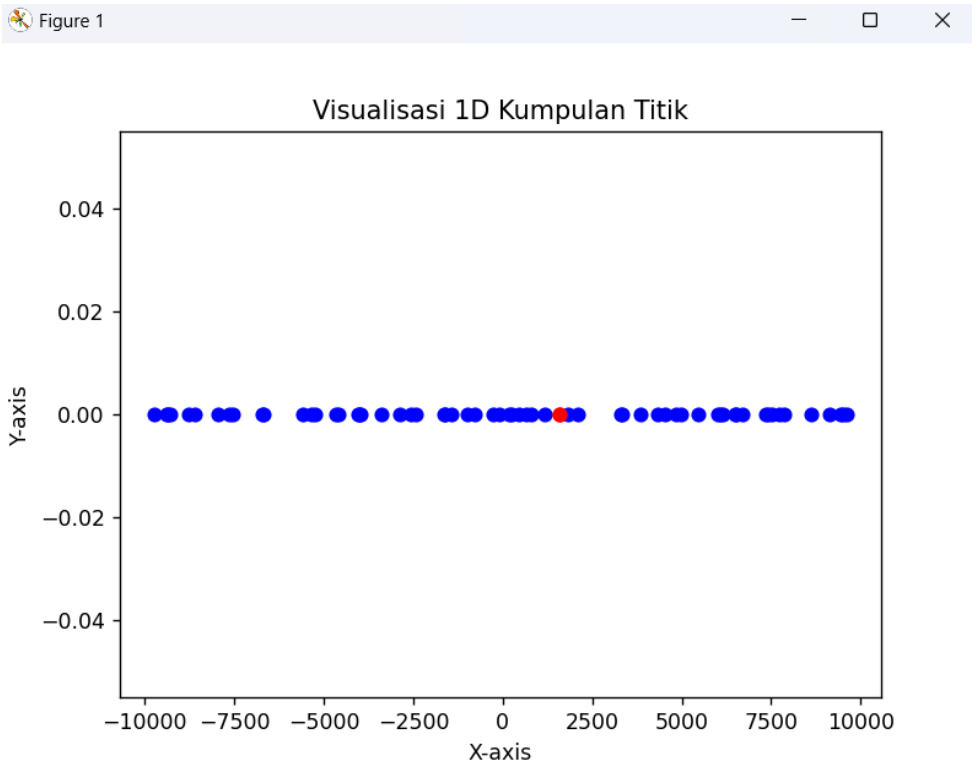
```
Masukkan banyak titik (n): 16
Masukkan dimensi titik (nDim): 10
Simpan output ke file ? Y/N
Y
Masukkan nama file : test16Dim10.txt
Divide and Conquer
Closest Points:  ( [-2048.2327937740856, -3094.8369860515368, -3094.996184214254, 6206.0885230918975, -5851.6913427
6919, 6195.839581792925, -6960.82586999271, 6822.065471957405, -818.730170210325, 1574.795447499664] , [-615.068703
4466646, -7622.327854313491, 4950.548163937314, 5311.245782974365, -5646.370347086933, 4019.735867832749, -7750.826
650107556, 8659.321978124513, -4820.53643793972, -5385.974326490759] )
Distance:  12701.481263922162
Brute Force
Closest Points:  ( [-615.0687034466646, -7622.327854313491, 4950.548163937314, 5311.245782974365, -5646.37034708693
3, 4019.735867832749, -7750.826650107556, 8659.321978124513, -4820.53643793972, -5385.974326490759] , [-2048.232793
```

```
7740856, -3094.8369860515368, -3094.996184214254, 6206.0885230918975, -5851.69134276919, 6195.839581792925, -6960.8
2586999271, 6822.065471957405, -818.730170210325, 1574.795447499664] )
Distance:  12701.481263922162
N perhitungan
Divide and Conquer:  58
Brute Force:  120
Execution Time
Divide and Conquer:  0.004000425338745117 s
Brute Force:  0.0030367374420166016 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```

3.2 n = 64

Dimensi 1

```
Masukkan banyak titik (n): 64
Masukkan dimensi titik (nDim): 1
Simpan output ke file ? Y/N
Y
Masukkan nama file : test64Dim1.txt
Divide and Conquer
Closest Points:  ( [1572.9716055172303] , [1575.3571030105522] )
Distance:  2.3854974933219637
Brute Force
Closest Points:  ( [1572.9716055172303] , [1575.3571030105522] )
Distance:  2.3854974933219637
N perhitungan
Divide and Conquer:  40
Brute Force:  2016
Execution Time
Divide and Conquer:  0.0010001659393310547 s
Brute Force:  0.012069940567016602 s
Computer Model
Zephyrus M GM501GM
```



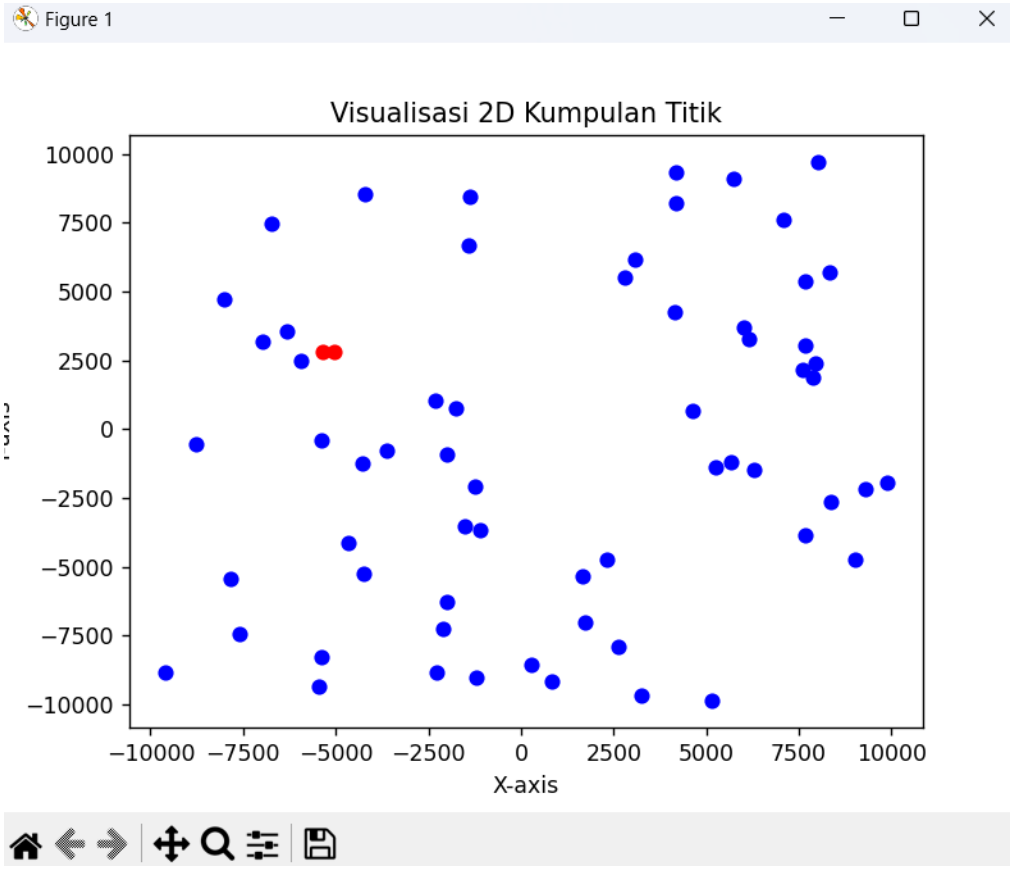
test64Dim1 - Notepad

File Edit View

```
Divide and Conquer
Closest Points: ([1572.9716055172303],[1575.3571030105522])
Distance: 2.3854974933219637
Brute Force
Closest Points: ([1572.9716055172303],[1575.3571030105522])
Distance: 2.3854974933219637
N perhitungan
Divide and Conquer: 40
Brute Force: 2016
Execution Time
Divide and Conquer: 0.0010001659393310547s
Brute Force: 0.012069940567016602s
Computer Model

Zephyrus M GM501GM
```

Dimensi 2

```
Masukkan banyak titik (n): 64
Masukkan dimensi titik (nDim): 2
Simpan output ke file ? Y/N
Y
Masukkan nama file : test64Dim2.txt
Divide and Conquer
Closest Points:  ( [-5349.341172137807, 2782.6714636766956] , [-5030.7515135772655, 2826.1320809613935] )
Distance:  321.54034862904746
Brute Force
Closest Points:  ( [-5349.341172137807, 2782.6714636766956] , [-5030.7515135772655, 2826.1320809613935] )
Distance:  321.54034862904746
N perhitungan
Divide and Conquer:  50
Brute Force:  2016
Execution Time
Divide and Conquer:  0.0009996891021728516 s
Brute Force:  0.011523246765136719 s
Computer Model
Zephyrus M GM501GM
```



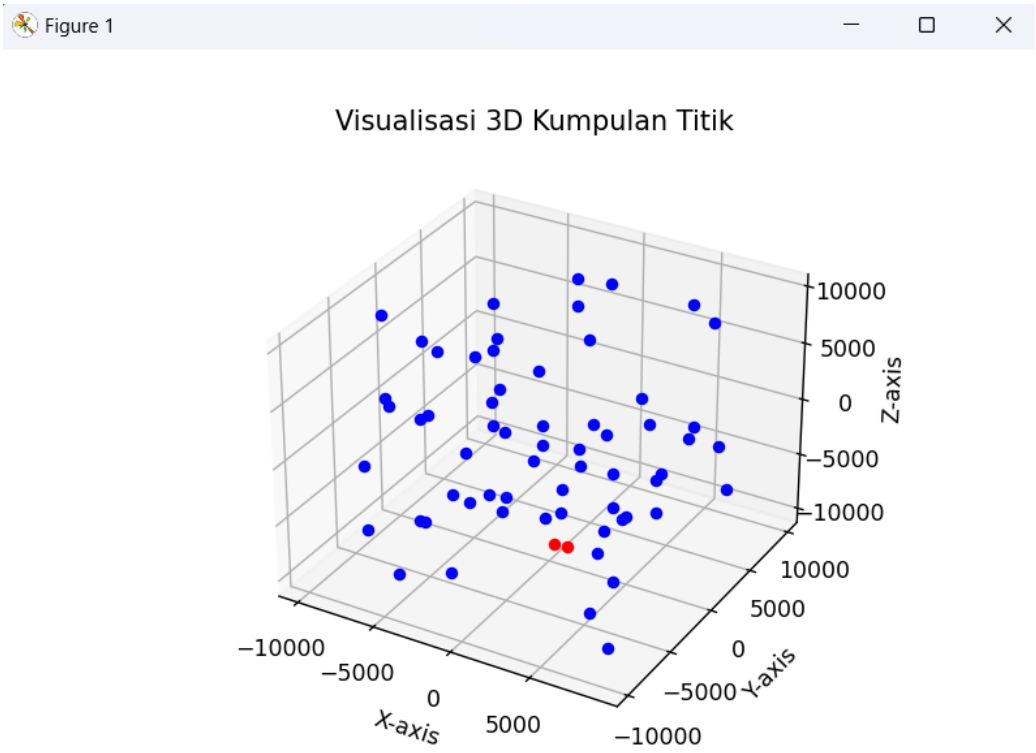
test64Dim2 - Notepad

File Edit View

```
Divide and Conquer
Closest Points: ([-5349.341172137807, 2782.6714636766956],[-5030.7515135772655, 2826.1320809613935])
Distance: 321.54034862904746
Brute Force
Closest Points: ([-5349.341172137807, 2782.6714636766956],[-5030.7515135772655, 2826.1320809613935])
Distance: 321.54034862904746
N perhitungan
Divide and Conquer: 50
Brute Force: 2016
Execution Time
Divide and Conquer: 0.0009996891021728516s
Brute Force: 0.011523246765136719s
Computer Model

Zephyrus M GM501GM
```

Dimensi 3

```
Masukkan banyak titik (n): 64
Masukkan dimensi titik (nDim): 3
Simpan output ke file ? Y/N
Y
Masukkan nama file : test64Dim3.txt
Divide and Conquer
Closest Points:  ( [4672.154663767318, -7898.912375359974, -2080.6026433722536] , [5650.641860434162, -8162.001633337834, -179
8.3950140551387] )
Distance:  1051.8052565703715
Brute Force
Closest Points:  ( [4672.154663767318, -7898.912375359974, -2080.6026433722536] , [5650.641860434162, -8162.001633337834, -179
8.3950140551387] )
Distance:  1051.8052565703715
N perhitungan
Divide and Conquer:  61
Brute Force:  2016
Execution Time
Divide and Conquer:  0.003988981246948242 s
Brute Force:  0.015023946762084961 s
Computer Model
Zephyrus M GM501GM
```



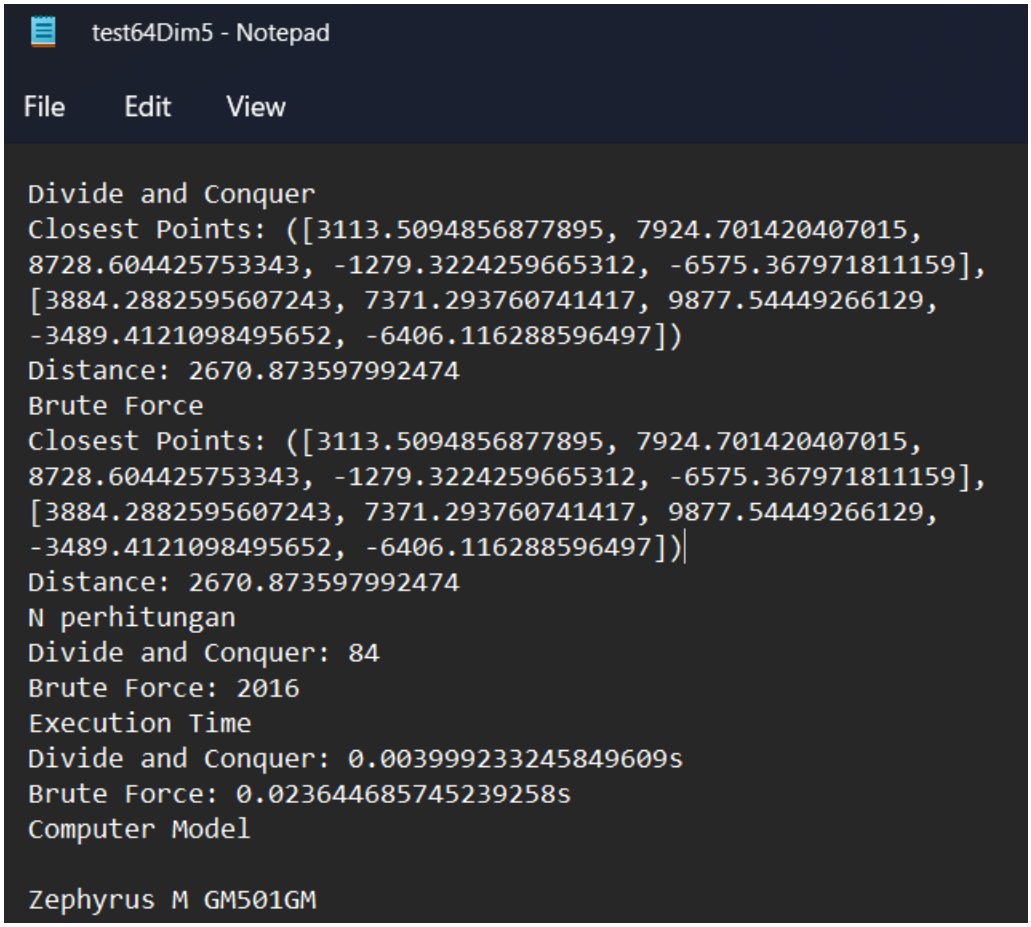
test64Dim3 - Notepad

File Edit View

```
Divide and Conquer
Closest Points: ([4672.154663767318, -7898.912375359974, -2080.6026433722536],[5650.641860434162, -8162.001633337834, -1798.3950140551387])
Distance: 1051.8052565703715
Brute Force
Closest Points: ([4672.154663767318, -7898.912375359974, -2080.6026433722536],[5650.641860434162, -8162.001633337834, -1798.3950140551387])
Distance: 1051.8052565703715
N perhitungan
Divide and Conquer: 61
Brute Force: 2016
Execution Time
Divide and Conquer: 0.003988981246948242s
Brute Force: 0.015023946762084961s
Computer Model

Zephyrus M GM501GM
```

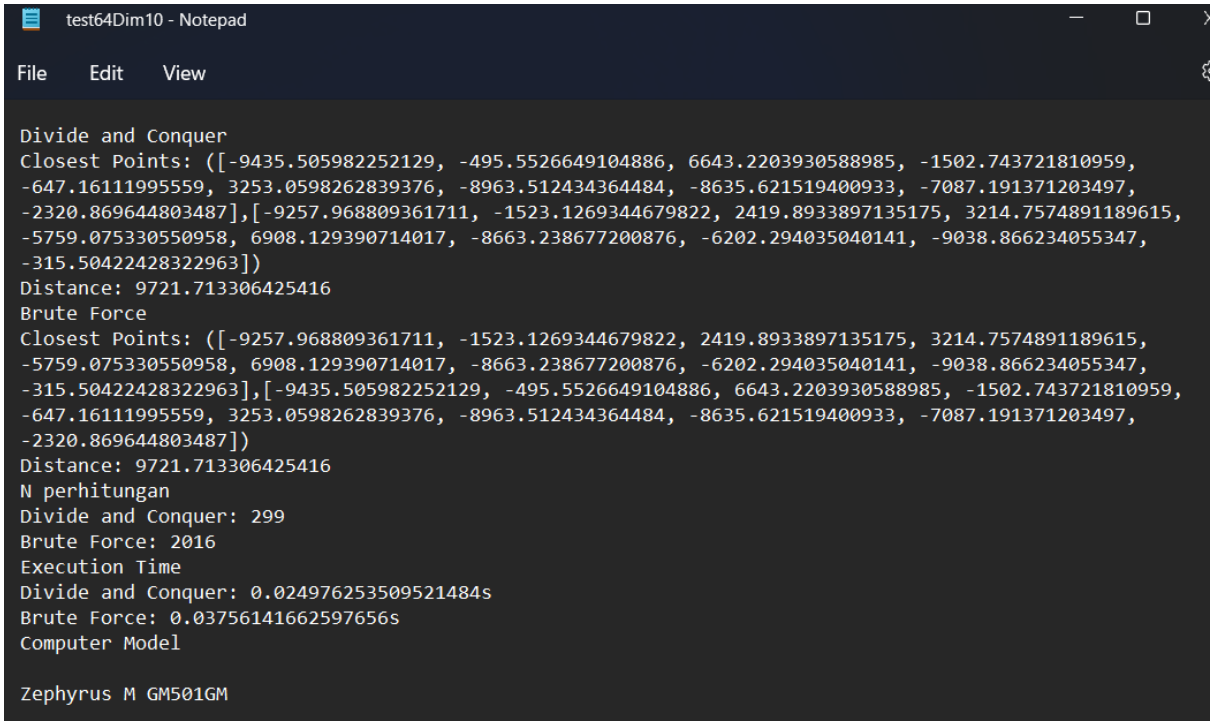
Dimensi 5

```
Masukkan banyak titik (n): 64
Masukkan dimensi titik (nDim): 5
Simpan output ke file ? Y/N
Y
Masukkan nama file : test64Dim5.txt
Divide and Conquer
Closest Points:  ( [3113.5094856877895, 7924.701420407015, 8728.604425753343, -1279.3224259665312, -6575.367971811159] , [3884
.2882595607243, 7371.293760741417, 9877.54449266129, -3489.4121098495652, -6406.116288596497] )
Distance:  2670.873597992474
Brute Force
Closest Points:  ( [3113.5094856877895, 7924.701420407015, 8728.604425753343, -1279.3224259665312, -6575.367971811159] , [3884
.2882595607243, 7371.293760741417, 9877.54449266129, -3489.4121098495652, -6406.116288596497] )
Distance:  2670.873597992474
N perhitungan
Divide and Conquer:  84
Brute Force:  2016
Execution Time
Divide and Conquer:  0.003999233245849609 s
Brute Force:  0.023644685745239258 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```



Dimensi 10

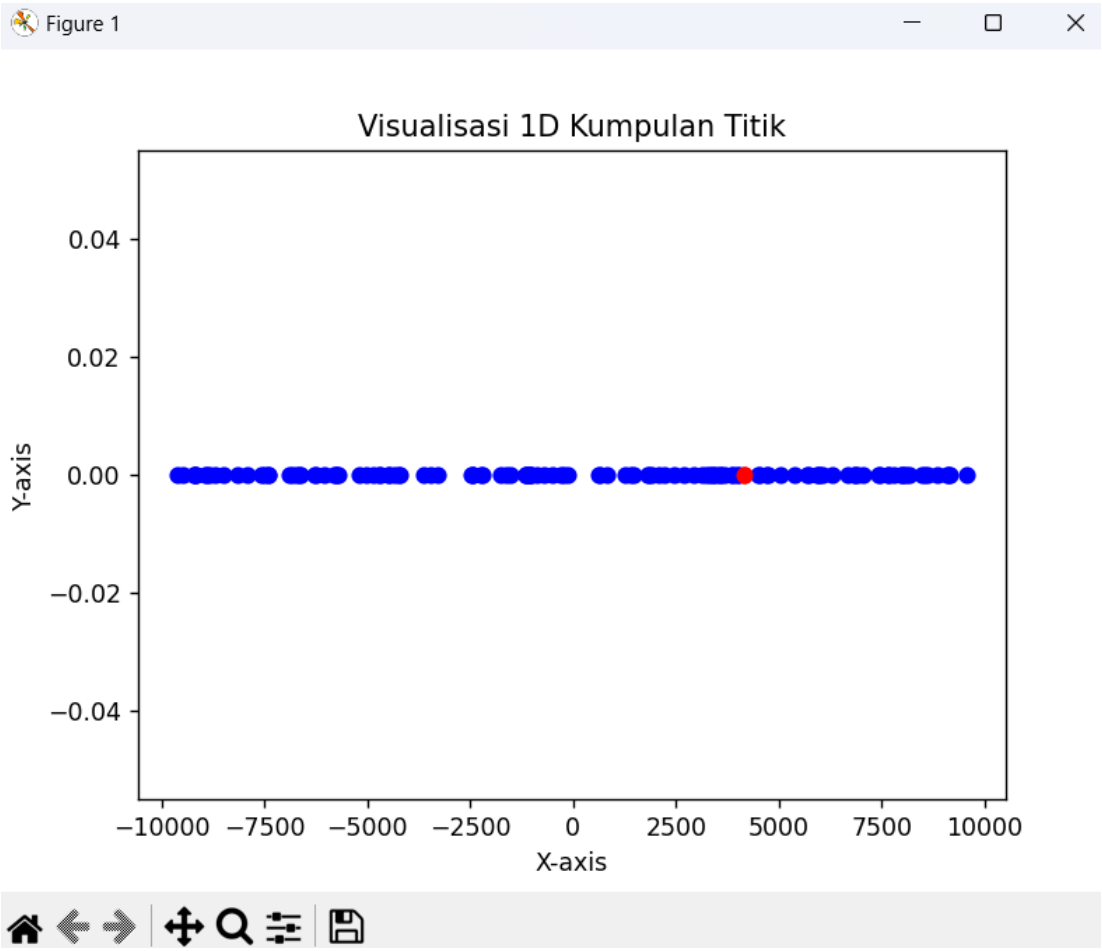
```
Masukkan banyak titik (n): 64
Masukkan dimensi titik (nDim): 10
Simpan output ke file ? Y/N
Y
Masukkan nama file : test64Dim10.txt
Divide and Conquer
Closest Points:  ( [-9435.505982252129, -495.5526649104886, 6643.2203930588985, -1502.743721810959, -647.16111995559, 3253.059
8262839376, -8963.512434364484, -8635.621519400933, -7087.191371203497, -2320.869644803487] , [-9257.968809361711, -1523.12693
44679822, 2419.8933897135175, 3214.7574891189615, -5759.075330550958, 6908.129390714017, -8663.238677200876, -6202.29403504014
1, -9038.866234055347, -315.50422428322963] )
Distance:  9721.713306425416
Brute Force
Closest Points:  ( [-9257.968809361711, -1523.1269344679822, 2419.8933897135175, 3214.7574891189615, -5759.075330550958, 6908.
129390714017, -8663.238677200876, -6202.294035040141, -9038.866234055347, -315.50422428322963] , [-9435.505982252129, -495.552
6649104886, 6643.2203930588985, -1502.743721810959, -647.16111995559, 3253.0598262839376, -8963.512434364484, -8635.6215194009
33, -7087.191371203497, -2320.869644803487] )
Distance:  9721.713306425416
N perhitungan
Divide and Conquer:  299
Brute Force:  2016
Execution Time
Divide and Conquer:  0.024976253509521484 s
Brute Force:  0.03756141662597656 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```



3.3 n = 128

Dimensi 1

```
Masukkan banyak titik (n): 128
Masukkan dimensi titik (nDim): 1
Simpan output ke file ? Y/N
Y
Masukkan nama file : test128Dim1.txt
Divide and Conquer
Closest Points:  ( [4154.285340581184] , [4154.319683165686] )
Distance:  0.034342584502155 66
Brute Force
Closest Points:  ( [4154.285340581184] , [4154.319683165686] )
Distance:  0.034342584502155 66
N perhitungan
Divide and Conquer:  74
Brute Force:  8128
Execution Time
Divide and Conquer:  0.0010013580322265625 s
Brute Force:  0.0420074462890625 s
Computer Model
Zephyrus M GM501GM
```



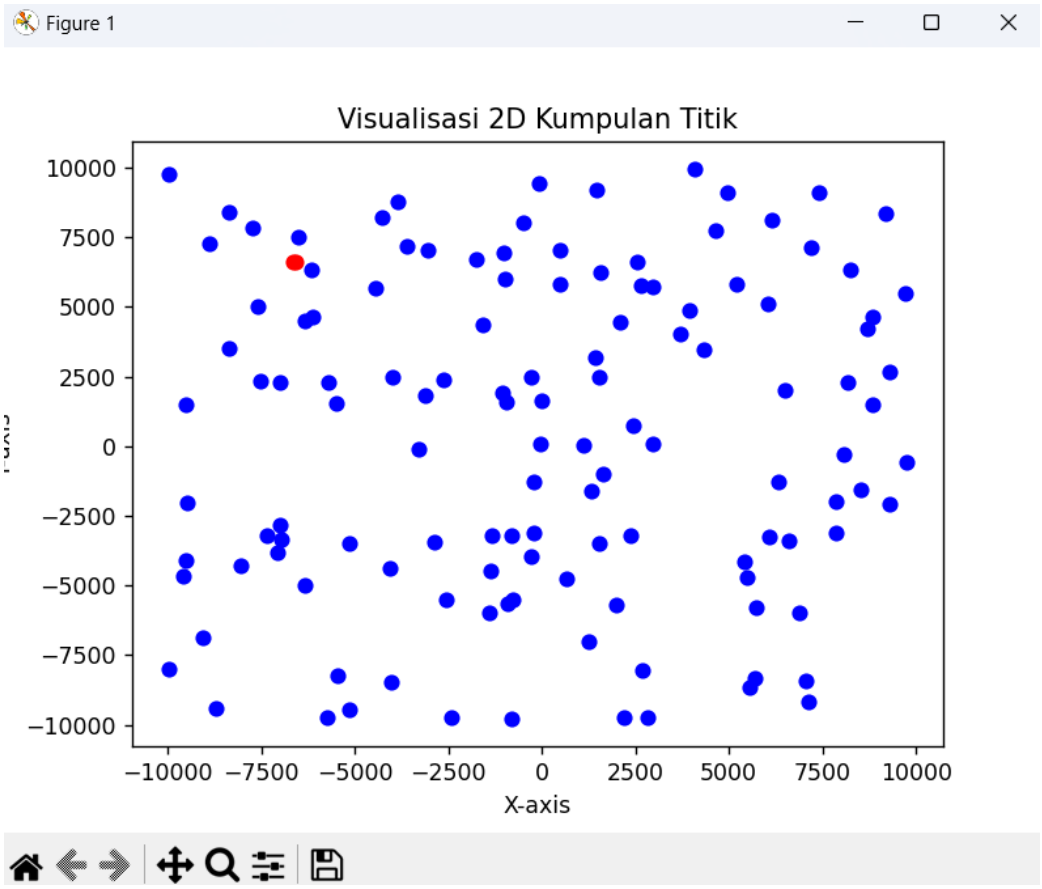
test128Dim1 - Notepad

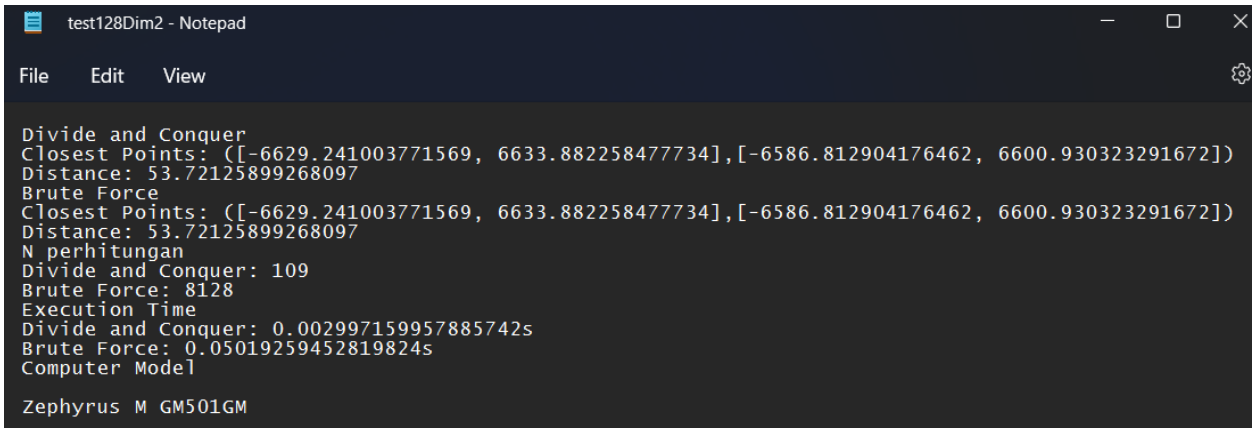
File Edit View

```
Divide and Conquer
Closest Points: ([4154.285340581184],[4154.319683165686])
Distance: 0.034342584502155 66
Brute Force
Closest Points: ([4154.285340581184],[4154.319683165686])
Distance: 0.034342584502155 66
N perhitungan
Divide and Conquer: 74
Brute Force: 8128
Execution Time
Divide and Conquer: 0.0010013580322265625s
Brute Force: 0.0420074462890625s
Computer Model

Zephyrus M GM501GM
```

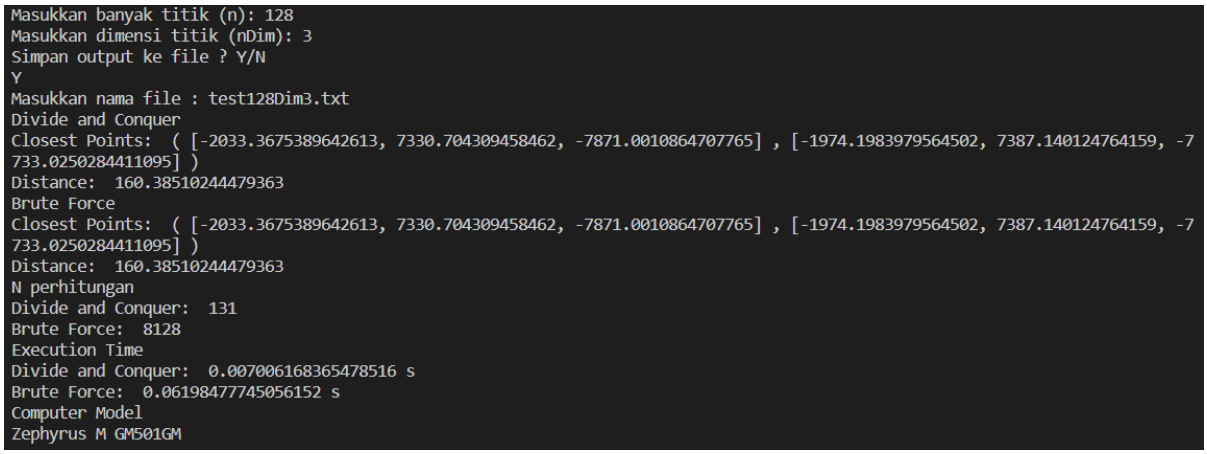
Dimensi 2

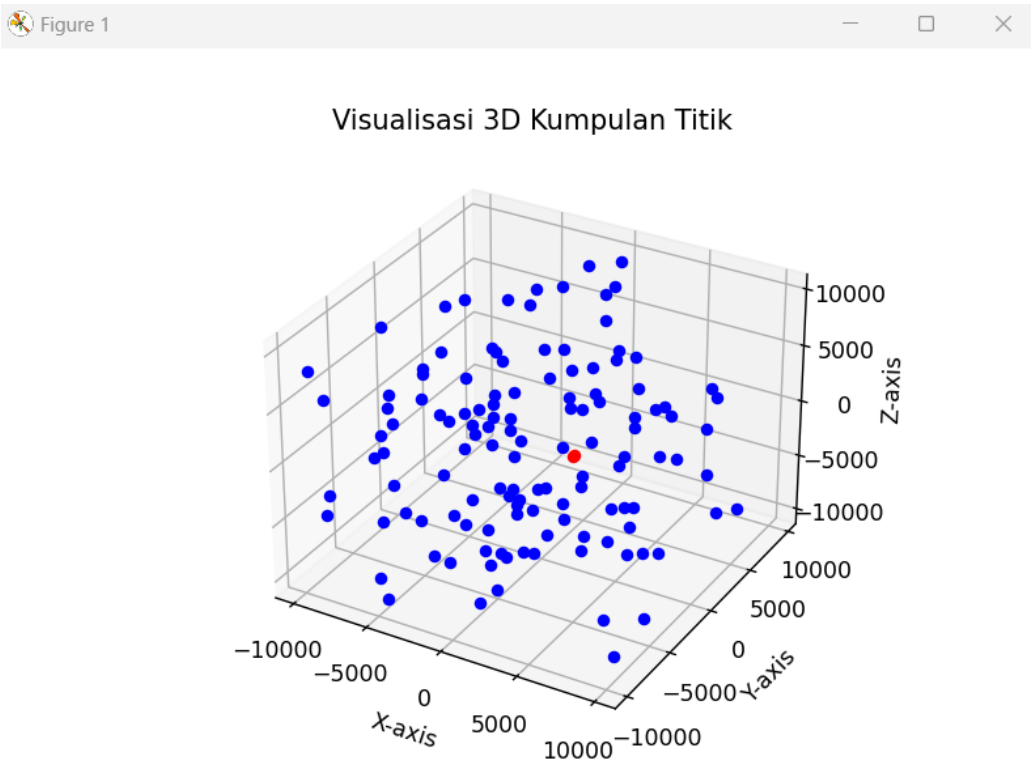
```
Masukkan banyak titik (n): 128
Masukkan dimensi titik (nDim): 2
Simpan output ke file ? Y/N
Y
Masukkan nama file : test128Dim2.txt
Divide and Conquer
Closest Points:  ( [-6629.241003771569, 6633.882258477734] , [-6586.812904176462, 6600.930323291672] )
Distance:  53.72125899268097
Brute Force
Closest Points:  ( [-6629.241003771569, 6633.882258477734] , [-6586.812904176462, 6600.930323291672] )
Distance:  53.72125899268097
N perhitungan
Divide and Conquer:  109
Brute Force:  8128
Execution Time
Divide and Conquer:  0.002997159957885742 s
Brute Force:  0.05019259452819824 s
Computer Model
Zephyrus M GM501GM
```




test128Dim2 - Notepad
File Edit View
Divide and Conquer
Closest Points: ([-6629.241003771569, 6633.882258477734],[-6586.812904176462, 6600.930323291672])
Distance: 53.72125899268097
Brute Force
Closest Points: ([-6629.241003771569, 6633.882258477734],[-6586.812904176462, 6600.930323291672])
Distance: 53.72125899268097
N perhitungan
Divide and Conquer: 109
Brute Force: 8128
Execution Time
Divide and Conquer: 0.002997159957885742s
Brute Force: 0.050192594528198245
Computer Model
Zephyrus M GM501GM

Dimensi 3


Masukkan banyak titik (n): 128
Masukkan dimensi titik (nDim): 3
Simpan output ke file ? Y/N
Y
Masukkan nama file : test128Dim3.txt
Divide and Conquer
Closest Points: ([-2033.3675389642613, 7330.704309458462, -7871.0010864707765] , [-1974.1983979564502, 7387.140124764159, -7733.0250284411095])
Distance: 160.38510244479363
Brute Force
Closest Points: ([-2033.3675389642613, 7330.704309458462, -7871.0010864707765] , [-1974.1983979564502, 7387.140124764159, -7733.0250284411095])
Distance: 160.38510244479363
N perhitungan
Divide and Conquer: 131
Brute Force: 8128
Execution Time
Divide and Conquer: 0.007006168365478516 s
Brute Force: 0.06198477745056152 s
Computer Model
Zephyrus M GM501GM



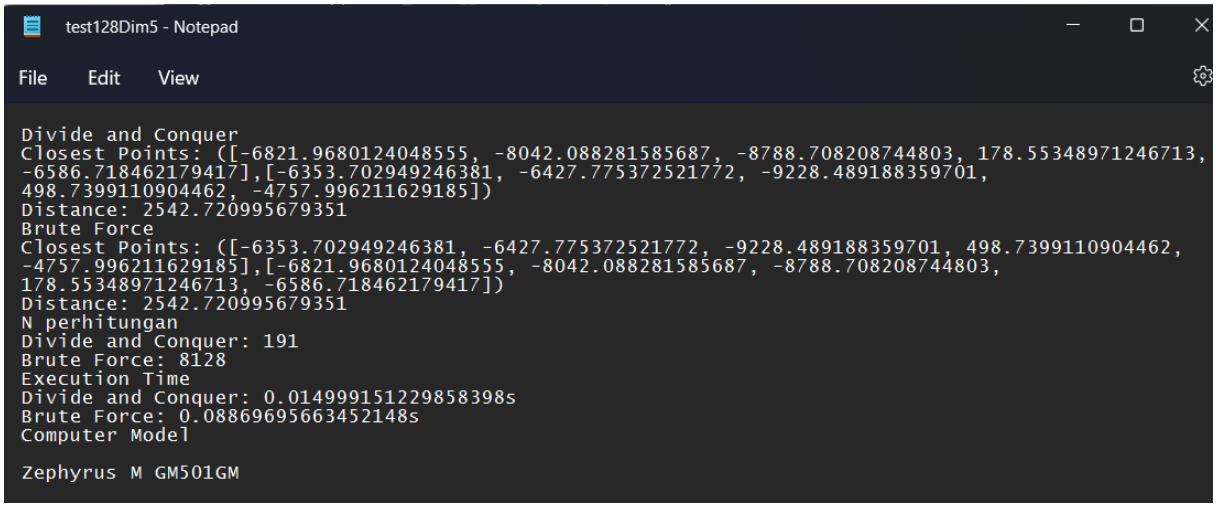
test128Dim3 - Notepad

File Edit View

```
Divide and Conquer
Closest Points: ([-2033.3675389642613, 7330.704309458462, -7871.0010864707765],
[-1974.1983979564502, 7387.140124764159, -7733.02502844110951])
Distance: 160.38510244479363
Brute Force
Closest Points: ([-2033.3675389642613, 7330.704309458462, -7871.0010864707765],
[-1974.1983979564502, 7387.140124764159, -7733.02502844110951])
Distance: 160.38510244479363
N perhitungan
Divide and Conquer: 131
Brute Force: 8128
Execution Time
Divide and Conquer: 0.007006168365478516s
Brute Force: 0.061984777450561525
Computer Model

Zephyrus M GM501GM
```

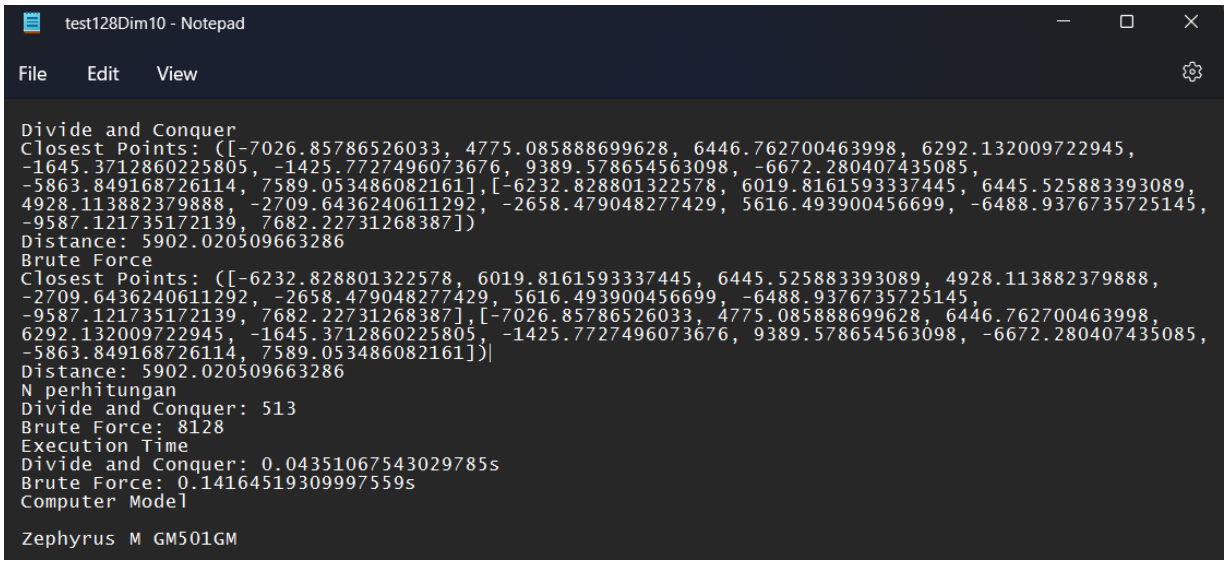
Dimensi 5

```
Masukkan banyak titik (n): 128
Masukkan dimensi titik (nDim): 5
Simpan output ke file ? Y/N
Y
Masukkan nama file : test128Dim5.txt
Divide and Conquer
Closest Points:  ( [-6821.9680124048555, -8042.088281585687, -8788.708208744803, 178.55348971246713, -6586.718462179417]
, [-6353.702949246381, -6427.775372521772, -9228.489188359701, 498.7399110904462, -4757.996211629185] )
Distance:  2542.720995679351
Brute Force
Closest Points:  ( [-6353.702949246381, -6427.775372521772, -9228.489188359701, 498.7399110904462, -4757.996211629185] ,
[-6821.9680124048555, -8042.088281585687, -8788.708208744803, 178.55348971246713, -6586.718462179417] )
Distance:  2542.720995679351
N perhitungan
Divide and Conquer:  191
Brute Force:  8128
Execution Time
Divide and Conquer:  0.014999151229858398 s
Brute Force:  0.08869695663452148 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```



Dimensi 10

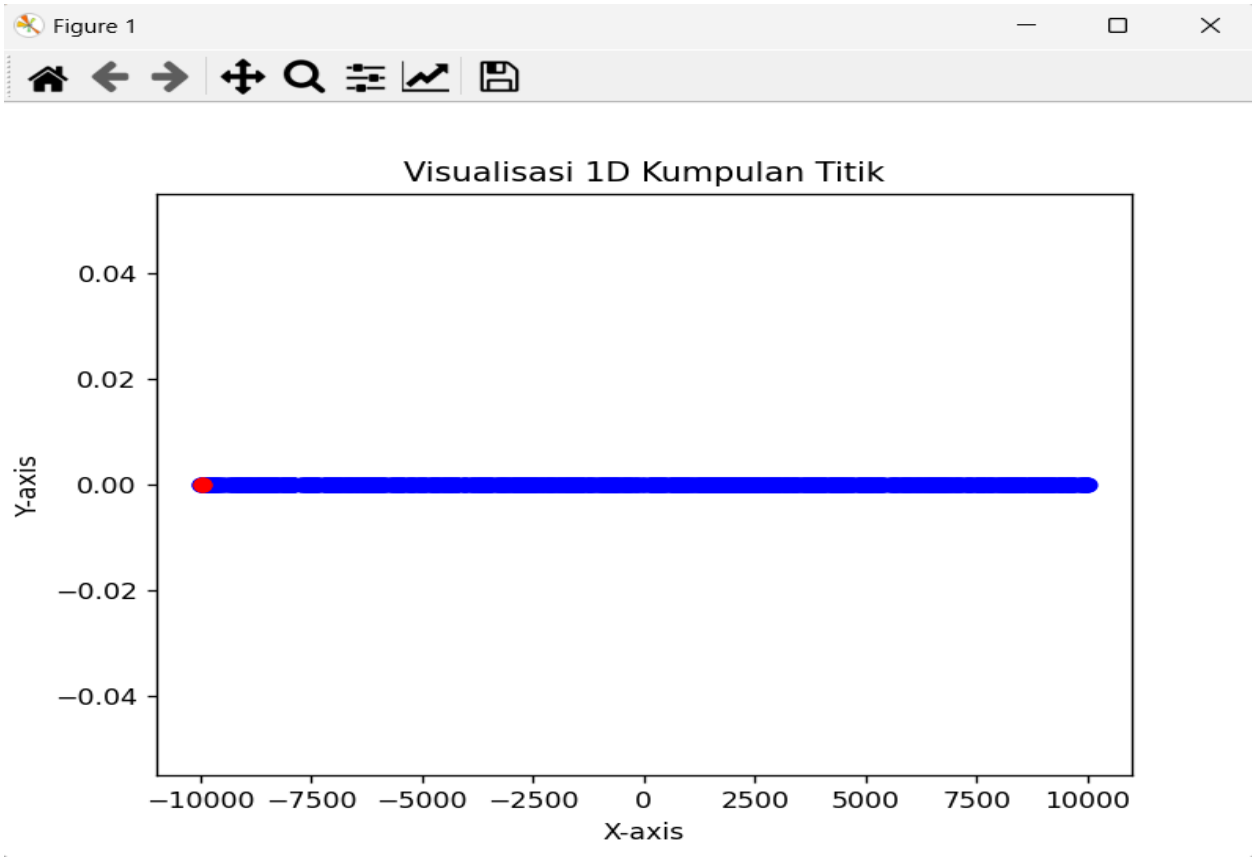
```
Masukkan banyak titik (n): 128
Masukkan dimensi titik (nDim): 10
Simpan output ke file ? Y/N
Y
Masukkan nama file : test128Dim10.txt
Divide and Conquer
Closest Points:  ( [-7026.85786526033, 4775.085888699628, 6446.762700463998, 6292.132009722945, -1645.3712860225805,
-1425.7727496073676, 9389.578654563098, -6672.280407435085, -5863.849168726114, 7589.053486082161] , [-6232.828801322
578, 6019.8161593337445, 6445.525883393089, 4928.113882379888, -2709.6436240611292, -2658.479048277429, 5616.49390045
6699, -6488.9376735725145, -9587.121735172139, 7682.22731268387] )
Distance:  5902.020509663286
Brute Force
Closest Points:  ( [-6232.828801322578, 6019.8161593337445, 6445.525883393089, 4928.113882379888, -2709.6436240611292
, -2658.479048277429, 5616.493900456699, -6488.9376735725145, -9587.121735172139, 7682.22731268387] , [-7026.85786526
033, 4775.085888699628, 6446.762700463998, 6292.132009722945, -1645.3712860225805, -1425.7727496073676, 9389.57865456
3098, -6672.280407435085, -5863.849168726114, 7589.053486082161] )
Distance:  5902.020509663286
N perhitungan
Divide and Conquer:  513
Brute Force:  8128
Execution Time
Divide and Conquer:  0.04351067543029785 s
Brute Force:  0.14164519309997559 s
Computer Model
Zephyrus M GM501GM


Kumpulan titik tidak dapat divisualisasikan
```



3.4 n = 1000

Dimensi 1

```
Masukkan banyak titik (n): 1000
Masukkan dimensi titik (nDim): 1
Simpan output ke file ? Y/N
Y
Masukkan nama file : test1000Dim1.txt
Divide and Conquer
Closest Points:  ( [-9940.992112923941] , [-9940.950714439963] )
Distance:  0.04139848397790047
Brute Force
Closest Points:  ( [-9940.950714439963] , [-9940.992112923941] )
Distance:  0.04139848397790047
N perhitungan
Divide and Conquer:  626
Brute Force:  499500
Execution Time
Divide and Conquer:  0.0019986629486083984 s
Brute Force:  0.6125226020812988 s
Computer Model
82FE
```

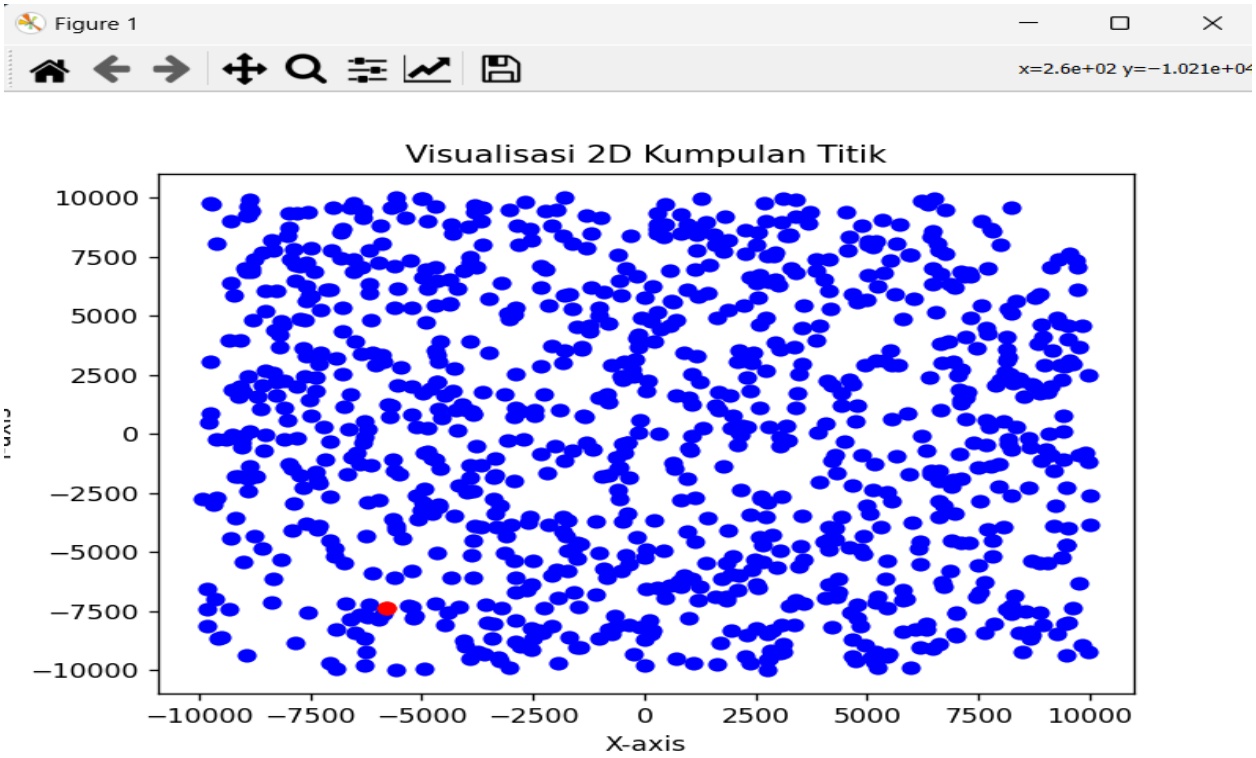


test1000Dim1.txt

```
Divide and Conquer
Closest Points: ([-9940.992112923941],[-9940.950714439963])
Distance: 0.04139848397790047
Brute Force
Closest Points: ([-9940.950714439963],[-9940.992112923941])
Distance: 0.04139848397790047
N perhitungan
Divide and Conquer: 626
Brute Force: 499500
Execution Time
Divide and Conquer: 0.0019986629486083984s
Brute Force: 0.6125226020812988s
Computer Model


82FE
```

Dimensi 2

```
Masukkan banyak titik (n): 1000
Masukkan dimensi titik (nDim): 2
Simpan output ke file ? Y/N
Y
Masukkan nama file : test1000Dim2.txt
Divide and Conquer
Closest Points:  ( [-5789.146899120045, -7388.291455226268] , [-5781.5484434328, -7382.212002707171] )
Distance:  9.731200941455487
Brute Force
Closest Points:  ( [-5781.5484434328, -7382.212002707171] , [-5789.146899120045, -7388.291455226268] )
Distance:  9.731200941455487
N perhitungan
Divide and Conquer:  881
Brute Force:  499500
Execution Time
Divide and Conquer:  0.007998943328857422 s
Brute Force:  0.8534209728240967 s
Computer Model
82FE
```

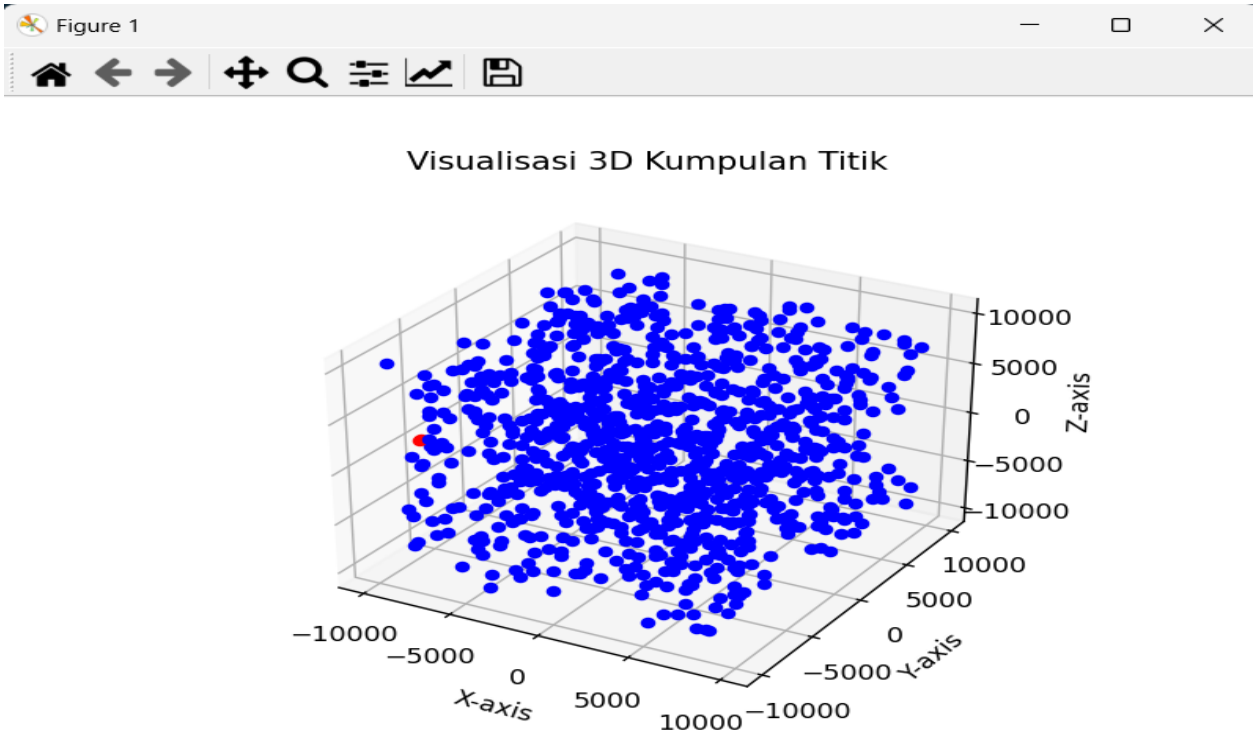


test1000Dim2.txt

```
Divide and Conquer
Closest Points: ([-5789.146899120045, -7388.291455226268],[-5781.5484434328, -7382.212002707171])
Distance: 9.731200941455487
Brute Force
Closest Points: ([-5781.5484434328, -7382.212002707171],[-5789.146899120045, -7388.291455226268])
Distance: 9.731200941455487
N perhitungan
Divide and Conquer: 881
Brute Force: 499500
Execution Time
Divide and Conquer: 0.007998943328857422s
Brute Force: 0.8534209728240967s
Computer Model


82FE
```

Dimensi 3

```
Masukkan banyak titik (n): 1000
Masukkan dimensi titik (nDim): 3
Simpan output ke file ? Y/N
Y
Masukkan nama file : test1000Dim3.txt
Divide and Conquer
Closest Points:  ( [-8077.475442189164, -8782.505310434839, 3124.8523125865686] , [-7967.381633285884, -8767.411459102841, 3178.469275860507] )
Distance:  123.38253465374746
Brute Force
Closest Points:  ( [-8077.475442189164, -8782.505310434839, 3124.8523125865686] , [-7967.381633285884, -8767.411459102841, 3178.469275860507] )
Distance:  123.38253465374746
N perhitungan
Divide and Conquer:  1025
Brute Force:  499500
Execution Time
Divide and Conquer:  0.02615213394165039 s
Brute Force:  1.0140330791473389 s
Computer Model
82FE
```



.

test1000Dim3.txt

```
Divide and Conquer
Closest Points: ([-8077.475442189164, -8782.505310434839, 3124.8523125865686],[-7967.381633285884, -8767.411459102841, 3178.469275860507])
Distance: 123.38253465374746
Brute Force
Closest Points: ([-8077.475442189164, -8782.505310434839, 3124.8523125865686],[-7967.381633285884, -8767.411459102841, 3178.469275860507])
Distance: 123.38253465374746
N perhitungan
Divide and Conquer: 1025
Brute Force: 499500
Execution Time
Divide and Conquer: 0.026152133941650395
Brute Force: 1.0140330791473389
Computer Model


82FE
```

Dimensi 5

```
Masukkan banyak titik (n): 1000
Masukkan dimensi titik (nDim): 5
Simpan output ke file ? Y/N
Y
Masukkan nama file : test1000Dim5.txt
Divide and Conquer
Closest Points:  ( [-656.6258792783901, -5743.5034005002535, 2534.6445573138535, -5106.580241806771, 4152.451819509339] , [-507.63843039564745, -6410.445793336053, 3041.1257137338616, -5142.4889
49614035, 3844.821936482174] )
Distance:  905.239281804746
Brute Force
Closest Points:  ( [-656.6258792783901, -5743.5034005002535, 2534.6445573138535, -5106.580241806771, 4152.451819509339] , [-507.63843039564745, -6410.445793336053, 3041.1257137338616, -5142.4889
49614035, 3844.821936482174] )
Distance:  905.239281804746
N perhitungan
Divide and Conquer:  1371
Brute Force:  499500
Execution Time
Divide and Conquer:  0.08287882804870605 s
Brute Force:  1.4383981227874756 s
Computer Model
82FE


Kumpulan titik tidak dapat divisualisasikan
```

test1000Dim5.txt

```
Divide and Conquer
Closest Points: ([-656.6258792783901, -5743.5034005002535, 2534.6445573138535, -5106.580241806771, 4152.451819509339],[-507.63843039564745, -6410.445793336053, 3041.12571
Distance: 905.239281804746
Brute Force
Closest Points: ([-656.6258792783901, -5743.5034005002535, 2534.6445573138535, -5106.580241806771, 4152.451819509339],[-507.63843039564745, -6410.445793336053, 3041.12571
Distance: 905.239281804746
N perhitungan
Divide and Conquer: 1371
Brute Force: 499500
Execution Time
Divide and Conquer: 0.08287882804870605s
Brute Force: 1.4383981227874756s
Computer Model


82FE
```

Dimensi 10

```
Masukkan banyak titik (n): 1000
Masukkan dimensi titik (nDim): 10
Simpan output ke file ? Y/N
Y
Masukkan nama file : test1000Dim10.txt
Divide and Conquer
Closest Points:  ( [1696.0364075661055, -3964.1361703502007, 5568.823072870982, 6129.946495410646, 3479.499285580794, 4730.955158538127, 1546.7551042777195, -9864.708727630303, 2779.178036433224
4, 8538.655880483231] , [4234.959042648645, -4821.402821584857, 4556.656765571161, 8034.55269084825, 2939.0314427086887, 5924.363665111874, 2440.0777554740343, -9036.047850585803, 3869.845111892
231, 9663.32174956103] )
Distance:  4181.925537258
Brute Force
Closest Points:  ( [4234.959042648645, -4821.402821584857, 4556.656765571161, 8034.55269084825, 2939.0314427086887, 5924.363665111874, 2440.0777554740343, -9036.047850585803, 3869.845111892231,
9663.32174956103] , [1696.0364075661055, -3964.1361703502007, 5568.823072870982, 6129.946495410646, 3479.499285580794, 4730.955158538127, 1546.7551042777195, -9864.708727630303, 2779.178036433222
44, 8538.655880483231] )
Distance:  4181.925537258
N perhitungan
Divide and Conquer:  5470
Brute Force:  499500
Execution Time
Divide and Conquer:  0.38243675231933594 s
Brute Force:  2.7005419731140137 s
Computer Model
82FE


Kumpulan titik tidak dapat divisualisasikan
```

test1000Dim10.txt

```
Divide and Conquer
Closest Points: ([1696.0364075661055, -3964.1361703502007, 5568.823072870982, 6129.946495410646, 3479.499285580794, 4730.955158538127
Distance: 4181.925537258
Brute Force
Closest Points: ([4234.959042648645, -4821.402821584857, 4556.656765571161, 8034.55269084825, 2939.0314427086887, 5924.363665111874,
Distance: 4181.925537258
N perhitungan
Divide and Conquer: 5470
Brute Force: 499500
Execution Time
Divide and Conquer: 0.38243675231933594s
Brute Force: 2.7005419731140137s
Computer Model


82FE
```

Analisis : Terlihat bahwa algoritma *divide and conquer* lebih cepat dibanding algoritma *brute force* karena algoritma *divide and conquer* melakukan perhitungan jarak dengan jumlah yang lebih sedikit. Namun, jumlah perhitungan dan waktu eksekusi algoritma *divide and conquer* juga dipengaruhi oleh jumlah dimensi titik. Semakin banyak sumbu yang perlu diperhatikan, semakin banyak pula jumlah perhitungan yang dilakukan. Hal ini tidak terjadi pada algoritma *brute force* yang jumlah perhitungannya konstan terhadap banyak dimensi.

Lampiran

Link *repository* github : https://github.com/Mehmed13/Tucil2_13521066_13521172

Tabel check list :

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada kesalahan.	✓	
2. Program berhasil running	✓	
3. Program dapat menerima masukan dan dan menuliskan luaran.	✓	
4. Luaran program sudah benar (solusi closest pair benar)	✓	
5. Bonus 1 dikerjakan	✓	
6. Bonus 2 dikerjakan	✓	