LAPORAN TUGAS KECIL II

IF2211 STRATEGI ALGORITMA

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Disusun oleh

Akmal Mahardika Nurwahyu Pratama 13521070

SEMESTER IV TAHUN 2022/2023
JURUSAN TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

BAB I
ALGORITME DEVIDE AND CONQUER

Algoritme *devide and conquer* adalah algoritme yang pemograman yang didasarkan pada rekrusif multi-cabang. berdasarkan bahasanya, *devide* artinya membagi dan *conquer* artinya mengatasi. Algoritma ini memiliki langkah :

1. **Divide**
   Membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama).
2. **Conquer**
   Memecahkan (menyelesaikan) masing-masing upa-masalah (secara rekursif).
3. **Combine**
   Menggabungkan solusi masing-masing masalah sehingga membentuk solusi masalah semula.

Adapun contoh persoalan yang dapat diselesaikan dengan algoritme *devide and conquer* :

1. Persoalan MinMaks (mencari nilai minimum dan nilai maksimum)
2. Menghitung perpangkatan
3. Persoalan pengurutan (sorting) – Mergesort dan Quicksort
4. Mencari sepasang titik terdekat (closest pair problem)
5. Convex Hull
6. Perkalian matriks
7. Perkalian bilangan bulat besar
8. Perkalian dua buah polinom

Pada Tucil 2, mahasiswa 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 titk P1 = (x1, y1, z1) dan P2 = (x2, y2, z2) dihitung dengan rumus Euclidean berikut:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

BAB II
SOURCE PROGRAM

Program dibuat dengan menggunakan bahasa python dengan menggunakan *library*

- time
- matplotlib
- numpy
- random
- math

Dengan menggunakan fungsi / prosedur:

File tools.py	
	Membuat array of Titik dengan ukuran n x n. Array 2D diisi dengan nilai random. Nilai elemen dalam array 2d mungkin ada yang sama
`randomizeDot2(n, dimension)`	Sama seperti randomizedot tapi pakai loop
`isCloser(dot1, dot2, close_dis)`	True jika jarak abs(dot1[i]-dot2[i]) < close_dis**2 untuk i 0..len(dot1)
`deanDistance(dot1, dot2)`	Menghitung Eucledian Distance antara 2 titik, n Dimensi
`mergeSort(arr_dot)`	Melakukan merge sort sebuah array of titik berdasarkan nilai x1
`euclideanDistance(dot1, dot2)`	Menghitung Eucledian Distance antara 2 titik, n Dimensi

```python
def randomizeDot(n, dimension) :
    """ ...
    return np.random.uniform(constant.MIN_RAND, constant.MAX_RAND, (n, dimension))
```

```python
def randomizeDot2(n, dimension) :
    """ same as randomizeDot but using for loop """      You, 19 hours ago • create
    list_point = np.zeros((n, dimension))
    for i in range(n) :
        for j in range(dimension) :
            list_point[i][j] = random.uniform(constant.MIN_RAND, constant.MAX_RAND)
    return list_point
```

```python
def isCloser(dot1, dot2, close_dis) :
    """ True jika jarak abs(dot1[i]-dot2[i]) < close_dis**2 untuk i 0..len(dot1) """
    if len(dot1) == 1 :
        return abs(dot1[0]-dot2[0]) < close_dis
    else :
        mid = len(dot1)//2
        return isCloser(dot1[:mid], dot2[:mid], close_dis) and isCloser(dot1[mid:], dot2
        [mid:], close_dis)
```

```python
def mergeSort(arr_dot) :
    """ Melakukan merge sort sebuah array of titik berdasarkan nilai x1 """

    if len(arr_dot) > 1:
        # Devide
        mid = len(arr_dot)//2
        L = arr_dot[:mid]
```

```
        R = arr_dot[mid:]
        mergeSort(L)
        mergeSort(R)

        # Counquer
        i = j = k = 0
        l_len = len(L)
        r_len = len(R)
        temp_arr = np.zeros((len(arr_dot), len(arr_dot[0])))

        while i < l_len and j < r_len :
            if L[i][0] < R[j][0]:
                temp_arr[k] = L[i]
                i += 1
            else:
                temp_arr[k] = R[j]
                j += 1
            k += 1

        while i < l_len :
            temp_arr[k] = L[i]
            i += 1
            k += 1

        while j < r_len :
            temp_arr[k] = R[j]
            j += 1
            k += 1

        for i in range(len(arr_dot)):
            arr_dot[i] = temp_arr[i]
```

File DnCTitik.py	
devidenConquer(arr_dot, nCal)	Mencari jarak terdekat antara 2 titik dengan devide and conquer jarak dicari dari array of titik
baseCaseDNC(arr_dot, nCal)	Basecase dari devidenConquer, Mencari jarak terdekat antara 2 titik dari array 3 atau 2 titik
closestPairStrip(left_arr, right_arr, close_dis, close_dots, mid, nCal)	Mencari Jarak titik terdekat antara 2 titik yang dibatasi garis khayalan berdasarkan divide and conquer
stripClose(l_arr, r_arr, mid, close_dis)	Mencari titik terdekat antara 2 titik yang dibatasi garis khayalan berdasarkan divide and conquer

```
def devidenConquer(arr_dot, nCal) :
    """ Mencari jarak terdekat antara 2 titik dengan devide and conquer
```

```
    jarak dicari dari array of titik """

    length_arr = len(arr_dot)

    if length_arr <= 3 :
        return baseCaseDNC(arr_dot, nCal)
    else :
        mid = length_arr//2
        left_arr = arr_dot[:mid]
        right_arr = arr_dot[mid:]

        dis1, dots1, nCal = devidenConquer(left_arr, nCal)
        dis2, dots2, nCal = devidenConquer(right_arr, nCal)

        close_dis = min(dis1, dis2)
        if close_dis == dis1 :
            close_dots = dots1
        else :
            close_dots = dots2

        dis_mid, dots_mid, nCal = closestPairStrip(left_arr, right_arr,
close_dis, close_dots, arr_dot[mid], nCal)

        close_dis = min(close_dis, dis_mid)
        if close_dis == dis_mid :
            close_dots = dots_mid
        return close_dis, close_dots, nCal
```

```
def baseCaseDNC(arr_dot, nCal) :
    """ Basecase dari devidenConquer | Mencari jarak terdekat antara 2 titik
dari array 3 atau 2 titik """
    dis = tools.euclideanDistance(arr_dot[0], arr_dot[1])
    nCal += 1
    dots = [arr_dot[0], arr_dot[1]]

    if (len(arr_dot) == 3) :
        # Cek jika ada 3 titik
        for i in range(2) :     # 0-2 dan 1-2
            temp_dis = tools.euclideanDistance(arr_dot[i], arr_dot[2])
            nCal += 1
            if temp_dis < dis :
                dis = temp_dis
                dots = [arr_dot[i], arr_dot[2]]


    return dis, dots, nCal
```

```
def closestPairStrip(left_arr, right_arr, close_dis, close_dots, mid, nCal) :

    strip_arr, n_left, n_right = stripClose(left_arr, right_arr, mid,
close_dis)
    dots = close_dots
    dis = close_dis

    # Titik kiri hanayan mengecek titik kanan garis khayalan dan sebaliknya
(dengan jarak < close_dis)
    for i in range(n_left) :
        for j in range(n_left, n_left+n_right) :
            if (tools.isCloser(strip_arr[i], strip_arr[j], dis)) :
                temp_dis = tools.euclideanDistance(strip_arr[i], strip_arr[j])
                nCal += 1
                if temp_dis < dis :
                    dis = temp_dis
                    dots = [strip_arr[i], strip_arr[j]]
    return dis, dots, nCal
```

File ioApp.py	
`inputHandle()`	Memanggil fungsi/prosedur *input*
`inputNandDimension()`	Menangani input dari pengguna berupa dimensi dan banyak titik
`outputHandle(min_distance, min_dots,nCal, time)`	Mencetak hasil perhitungan program
`printTitik(arr_dot)`	mencetak pasangan titik dalam bentuk (x1, y1,...) dan (x2, y2,...)
`StartScreen()`	Spalsh dari Startscreen
`BoxOpenScreen(name)`	Box pembatas (buka)
`BoxCloseScreen(name)`	Box pembatas (tutup)

```
def inputHandle() :
    n, dimension = inputNandDimension()
    # n, dimendion = fileinput()
    return n, dimension
      def inputNandDimension():

    """ Menangani input dari pengguna berupa dimensi dan banyak titik]"""
    while (True) :
        n = input("Masukkan jumlah titik  : ")
        try :
            n = int(n)

            if n < 2 :
                print(dc.B_Red + f"Jumlah titik {dc.Underline}minimal 2!" +
dc.Reset)
                continue
            else :
                break
```

```
        except ValueError :
            print(dc.B_Red + "Input harus berupa angka!" + dc.Reset)
            continue

    while (True) :
        dimension = input("Masukkan dimensi titik : ")
        try :
            dimension = int(dimension)

            if dimension < 1 :
                print(dc.B_Red + f"Dimensi {dimension} tidak terdefini!
{dc.Underline}dimensi minimal 1!" + dc.Reset)
            else :
                break
        except ValueError :
            print(dc.B_Red + "Input harus berupa angka!" + dc.Reset)
            continue
    return n, dimension
def outputHandle(min_distance, min_dots, nCal, time) :
    """ Mencetak hasil perhitungan program """
    print("Jarak terdekat adalah              :", min_distance)
    print("Pasangan Titik terdekat adalah     :", end=' ')

    # Print pasangan titik
    printTitik(min_dots)

    print("Jumlah perhitungan jarak euclidean :", nCal)
    print("Waktu perhitungan                  :", time, "milidetik")
def printTitik(arr_dot) :
    """ mencetak pasangan titik dalam bentuk (x1, y1,...) dan (x2, y2,...) """
    for i in range(len(arr_dot)) :
        point = arr_dot[i]
        print(f"( {point[0] :.4f}", end = "")
        for j in range(1, len(point)):
            print(f", {point[j] :.4f}", end = "")
        print(" )", end = "")

        if i % 2 == 0:
            print(dc.B_Green+" dan ", end = "" + dc.Reset)
        else :
            print()
def StartScreen() :
    """ Spalsh dari Startscreen """
    print(dc.B_Magenta,"""
 $$$$$$\ $$\                                  $$\           $$$$$$$\ $$\
        $$\
$$  __$$\$$ |                                 $$ |          $$  __$$\\__|
         $$ |
```

```
$$ /  \__$$ |$$$$$$\  $$$$$$$\ $$$$$$\  $$$$$$$\$$$$$$\         $$ |  $$ $$\ $
$$$$$$\$$$$$$\   $$$$$$\ $$$$$$$\  $$$$$$$\ $$$$$$\
$$ |     $$ $$  __$$\$$  _____$$  __$$\$$  _____\_$$  _|        $$ |  $$ $$ $$
 _____\_$$  _|  \____$$\$$  __$$\$$  _____$$  __$$\
$$ |     $$ $$ /  $$ \$$$$$$\ $$$$$$$ \$$$$$$\   $$ |          $$ |  $$ $$ \$
$$$$$\   $$ |    $$$$$$ $$ |  $$ $$ /     $$$$$$$ |
$$ |  $$\$$ $$ |  $$ |\____$$\$$   ____|\____$$\  $$ |$$\       $$ |  $$ $$ |\
____$$\  $$ |$$\$$  __$$ $$ |  $$ $$ |     $$   ____|
\$$$$$$  $$ \$$$$$$  $$$$$$$  \$$$$$$$\$$$$$$$  |
\$$$$  |      $$$$$$$  $$ $$$$$$$  |
\$$$$  \$$$$$$$ $$ |  $$ \$$$$$$$\\$$$$$$$\
 \______/\__|\______/\_______/
\_______\_______/   \____/       \_______/\__\_______/   \____/
\_______\__|  \__|\_______|\_______|
          """, dc.Reset)
    print("{: ^144}".format(dc.B_Yellow + dc.Underline + "Akmal Mahardika
Nurwahyu Pratama - 13521070") + dc.Reset + "\n")

def BoxOpenScreen(name) :
    """ Box pembatas (buka) """
    print(dc.Bold + "┌{}{:^24}{}┐".format("─"*53,name,"─"*53) + dc.Reset +
"\n")

def BoxCloseScreen(name) :
    """ Box pembatas (tutup) """
    print("\n" + dc.Bold + "└{}{:^24}{}┘".format("─"*53,name,"─"*53) +
dc.Reset + "\n")
```

File bruteforce.py	
bruteforceDots(arr_dot)	Mencari pasangan titk terdekat dengan algoritma brute force

```
def bruteforceDots(arr_dot) :
    """ Mencari pasangan titk terdekat dengan algoritma brute force """
    nCal = 0
    dis = -1
    close_dis = 99999
    close_dots = []

    for i in range(len(arr_dot)) :
        for j in range(i+1, len(arr_dot)) :
            dis = tools.euclideanDistance(arr_dot[i], arr_dot[j])
            nCal += 1
            if dis < close_dis :
                close_dis = dis
                close_dots = [arr_dot[i], arr_dot[j]]
```

```
    return close_dis, close_dots, nCal
```

File visualizer.py	
visualize(arr_dots, min_dots)	Visualisasi array of titik
OneDPlot(arr_dots, min_dots)	Plotting 1D array of Titik
TwoDPlot(arr_dots, min_dots)	Plotting 2D array of Titik
ThreeDPlot(arr_dots, min_dots)	Plotting 3D array of Titik
connectDots(arr_dots, dim)	Menghubungkan pasangan titik yang ada di dalam array

```
def visualize(arr_dots, min_dots) :
    """ Visualisasi array of Titik """
    arr_dots = np.array(arr_dots)
    min_dots = np.array(min_dots)
    dim = len(arr_dots[0])
    if dim == 1 :
        OneDPlot(arr_dots, min_dots)
        return True
    elif dim == 2 :
        TwoDPlot(arr_dots, min_dots)
        return True
    elif dim == 3 :
        ThreeDPlot(arr_dots, min_dots)
        return True
    else :
        print(dc.B_Red + "Tidak dapat melakukan visualisasi" + dc.Reset)
        return False


def OneDPlot(arr_dots, min_dots) :
    """ Plotting 1D array of Titik """
    ax = plt.gca()
    ax.set_title("Visualisasi Titik")
    plt.plot(arr_dots, np.zeros_like(arr_dots), 'bo')
    plt.plot(min_dots, np.zeros_like(min_dots), 'ro')
    connectDots(min_dots,1)
    plt.xlabel('X')
    plt.show()

def TwoDPlot(arr_dots, min_dots) :
    """ Plotting 2D array of Titik """
    ax = plt.gca()
    ax.set_title("Visualisasi Titik")

    plt.scatter(arr_dots[:,0], arr_dots[:,1], marker='o', color='b')
    plt.scatter(min_dots[:,0], min_dots[:,1], marker='o', color='r')
    connectDots(min_dots,2)
```

```
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()

def ThreeDPlot(arr_dots, min_dots) :
    """ Plotting 3D array of Titik """
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')
    ax.set_title("Visualisasi Titik")
    ax.scatter(arr_dots[:,0], arr_dots[:,1], arr_dots[:,2], color='b')
    ax.scatter(min_dots[:,0], min_dots[:,1], min_dots[:,2], color='r')
    connectDots(min_dots,3, ax)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    plt.show()

def connectDots(arr_dots, dim, ax = None) :
    """ Menghubungkan pasangan titik yang ada di dalam array """
    if dim == 1 :
        for i in range(0, len(arr_dots), 2) :
            plt.plot([arr_dots[i], arr_dots[i+1]], [0,0], 'r-')
    elif dim == 2 :
        for i in range(0, len(arr_dots), 2) :
            plt.plot([arr_dots[i,0], arr_dots[i+1,0]], [arr_dots[i,1],
arr_dots[i+1,1]], 'r-')
    elif dim == 3 :
        # fig = plt.figure()
        # ax = fig.add_subplot(projection='3d')
        for i in range(0, len(arr_dots), 2) :
            ax.plot([arr_dots[i,0], arr_dots[i+1,0]], [arr_dots[i,1],
arr_dots[i+1,1]], [arr_dots[i,2], arr_dots[i+1,2]], 'r-')
```

Adapun file lain berisi atribut-atribut yang melengkapi program, beberapa diantaranya

- File constant.py

```
# declare constant
MAX_RAND = 500
MIN_RAND = -500
```

- File designCli.py

```
# Text Colour

Black = "\u001b[0;30m"
Red = "\u001b[0;31m"
Green = "\u001b[0;32m"
Yellow = "\u001b[0;33m"
```

```
Blue = "\u001b[0;34m"
Magenta = "\u001b[0;35m"
Cyan = "\u001b[0;36m"
White = "\u001b[0;37m"
Reset = "\u001b[0;00m"

B_Black = "\u001b[30;1m"
B_Red = "\u001b[31;1m"
B_Green = "\u001b[32;1m"
B_Yellow = "\u001b[33;1m"
B_Blue = "\u001b[34;1m"
B_Magenta = "\u001b[35;1m"
B_Cyan = "\u001b[36;1m"
B_White = "\u001b[37;1m"
Reset = "\u001b[0m"

# bg colour
Bg_Black = "\u001b[40m"
Bg_Red = "\u001b[41m"
Bg_Green = "\u001b[42m"
Bg_Yellow = "\u001b[43m"
Bg_Blue = "\u001b[44m"
Bg_Magenta = "\u001b[45m"
Bg_Cyan = "\u001b[46m"
Bg_White = "\u001b[47m"

# Text Style
Bold = "\u001b[1m"
Underline = "\u001b[4m"
Reversed = "\u001b[7m"
```

File -file diatas membentuk main program sebagai berikut (main.py)

```
# File : main.py
# Application to find nearest 2 point from n point in 3D space and calculate
distance between them

import time
import bruteforce as bf
import tools as tl
import DnCTitik as DnC
import ioApp as io
import visualizer as vis

if __name__ == "__main__":
    io.StartScreen()

    n, dimension = io.inputHandle()
    arr_dots = tl.randomizeDot2(n, dimension)
```

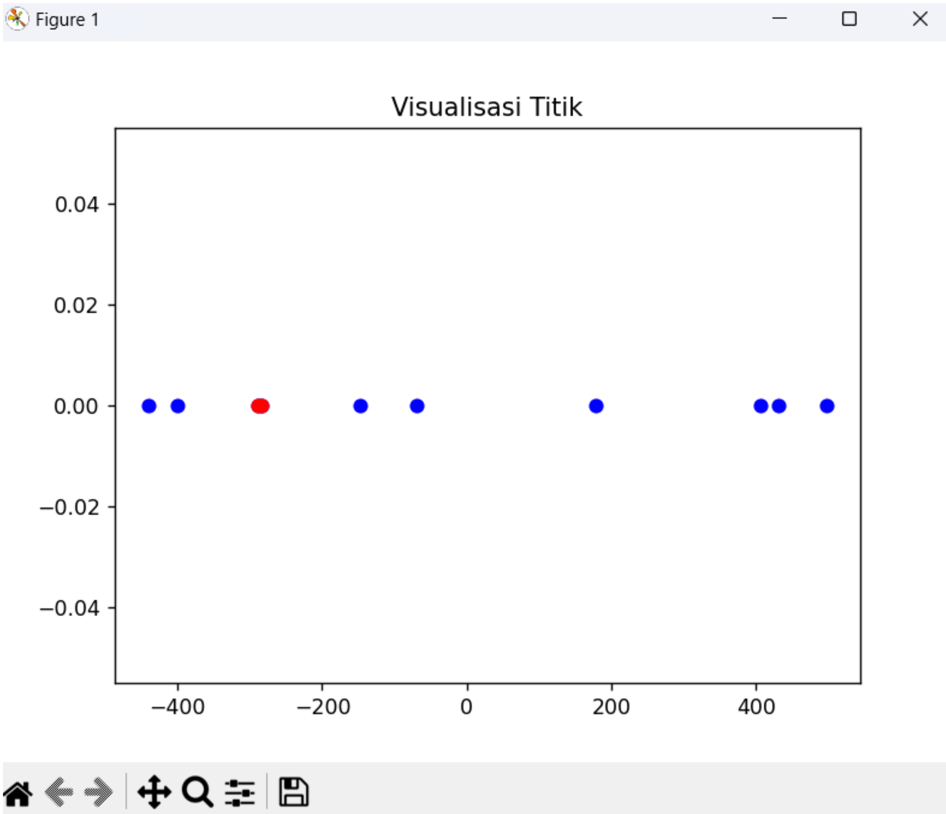
```
    tl.mergeSort(arr_dots)


    dnc_nCal = 0
    dnc_timeS = time.time()
    dnc_min_dis, dnc_min_dots, dnc_nCal  = DnC.devidenConquer(arr_dots,
dnc_nCal)
    dnc_timeE = time.time()

    io.BoxOpenScreen("Devide and Conquer")
    io.outputHandle(dnc_min_dis, dnc_min_dots, dnc_nCal, (dnc_timeE -
dnc_timeS)*1000)
    io.BoxCloseScreen("Devide and Conquer")
    vis.visualize(arr_dots, dnc_min_dots)

    bf_nCal = 0
    bf_timeS = time.time()
    bf_min_dis, bf_min_dots, bf_nCal  = bf.bruteforceDots(arr_dots)
    bf_timeE = time.time()

    io.BoxOpenScreen("Brute Force")
    io.outputHandle(bf_min_dis, bf_min_dots, bf_nCal, (bf_timeE -
bf_timeS)*1000)
    io.BoxCloseScreen("Brute Force")
    vis.visualize(arr_dots, bf_min_dots)
```

BAB III
TEST PROGRAM

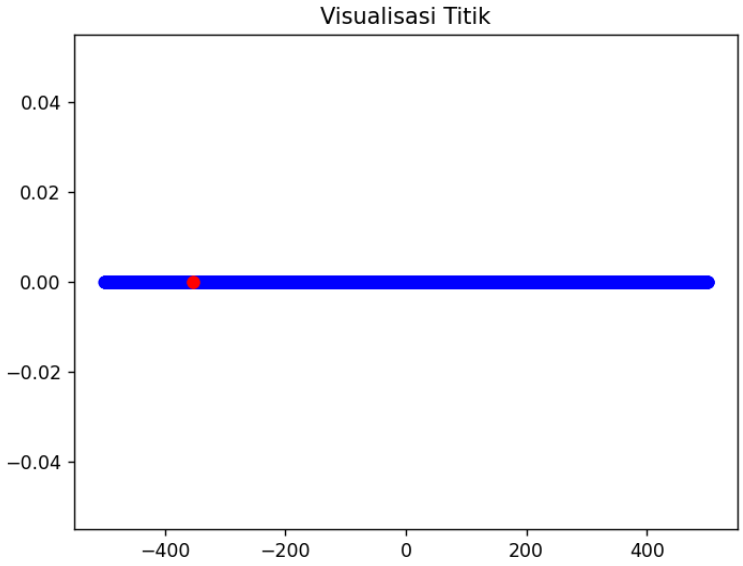
Input program terdiri dari 2 yaitu, jumlah titik (n) dan dimensi titik (d). Berikut merupakan beberapa test program yang dilakukan, test program dilakukan pada prosesor Intel i7-11567G dengan 8 GB Memori :

1. Test input keyboard benar untuk n = 10 dan d = 1

```
Akmal Mahardika Nurwahyu Pratama - 13521070

Masukkan jumlah titik  : 10
Masukkan dimensi titik : 1
┌──────────────────── Devide and Conquer ────────────────────┐

Jarak terdekat adalah              : 4.723404187559765
Pasangan Titik terdekat adalah     :
( -289.1895 ) dan ( -284.4661 )
Jumlah perhitungan jarak euclidean : 8
Waktu perhitungan                  : 0.0 milidetik

└──────────────────── Devide and Conquer ────────────────────┘
┌──────────────────── Brute Force ────────────────────┐

Jarak terdekat adalah              : 4.723404187559765
Pasangan Titik terdekat adalah     :
( -289.1895 ) dan ( -284.4661 )
Jumlah perhitungan jarak euclidean : 45
Waktu perhitungan                  : 0.0 milidetik

└──────────────────── Brute Force ────────────────────┘
```

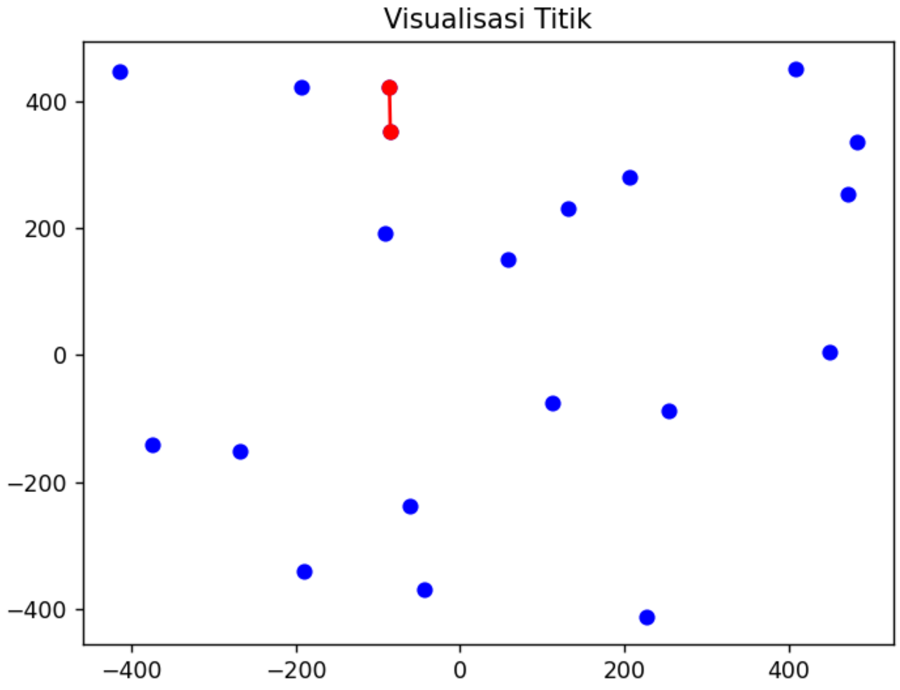


2. Test input keyboard benar untuk n = 5000 dan d = 1



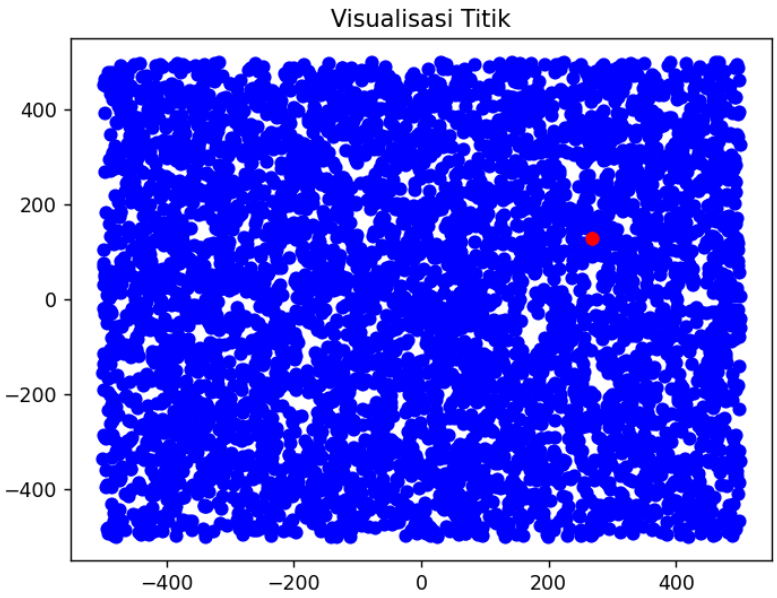
3. Test input keyboard benar untuk n = 20 dan d = 2

```
Masukkan jumlah titik  : 20
Masukkan dimensi titik : 2
┌──────────────────────────────────────────────────────   Devide and Conquer   ──────────────────────────────────────────────────────┐

Jarak terdekat adalah               : 68.96562589955319
Pasangan Titik terdekat adalah      : ( -85.7988, 421.8844 ) dan ( -84.7446, 352.9268 )
Jumlah perhitungan jarak euclidean : 19
Waktu perhitungan                   : 0.0 milidetik

└──────────────────────────────────────────────────────   Devide and Conquer   ──────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────      Brute Force       ──────────────────────────────────────────────────────┐

Jarak terdekat adalah               : 68.96562589955319
Pasangan Titik terdekat adalah      : ( -85.7988, 421.8844 ) dan ( -84.7446, 352.9268 )
Jumlah perhitungan jarak euclidean : 190
Waktu perhitungan                   : 0.0 milidetik

└──────────────────────────────────────────────────────      Brute Force       ──────────────────────────────────────────────────────┘
```

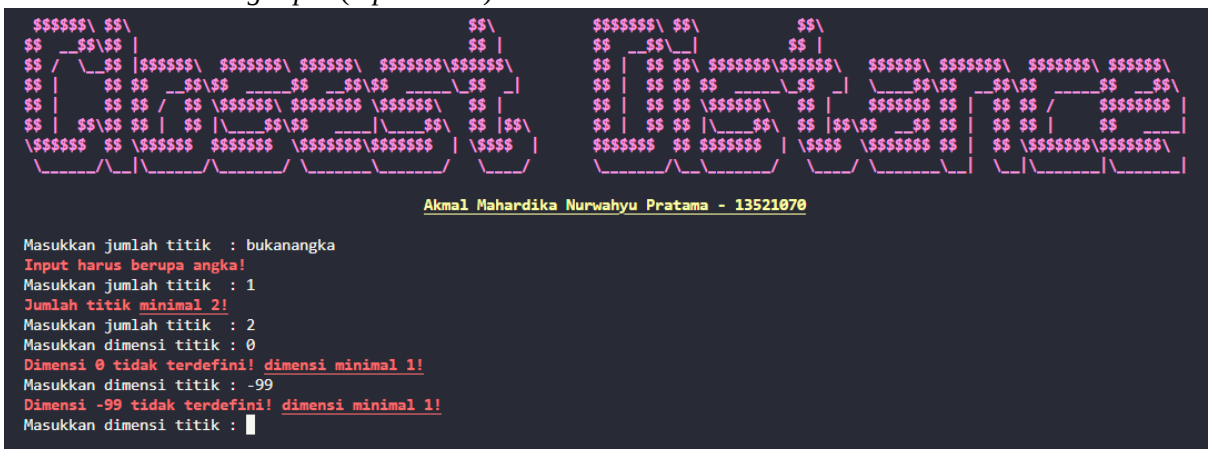


4. Testinput keyboard untuk n = 5000 dan d = 5

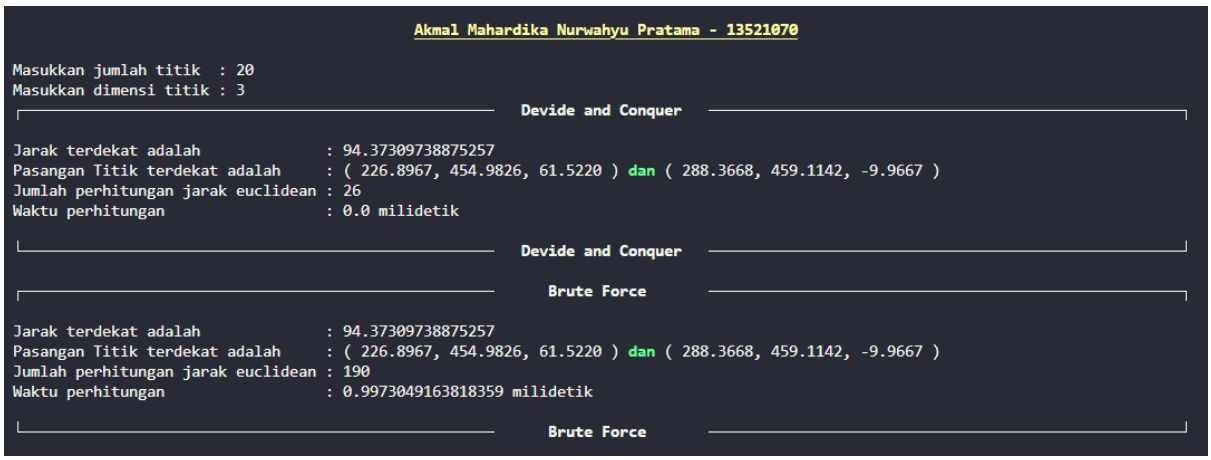
```
Masukkan jumlah titik  : 5000
Masukkan dimensi titik : 2
┌──────────────────────────────────────────────────────   Devide and Conquer   ──────────────────────────────────────────────────────┐

Jarak terdekat adalah               : 0.13114688554016798
Pasangan Titik terdekat adalah      : ( 267.1248, 128.5897 ) dan ( 267.1485, 128.7187 )
Jumlah perhitungan jarak euclidean : 5324
Waktu perhitungan                   : 75.11663436889648 milidetik

└──────────────────────────────────────────────────────   Devide and Conquer   ──────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────      Brute Force       ──────────────────────────────────────────────────────┐

Jarak terdekat adalah               : 0.13114688554016798
Pasangan Titik terdekat adalah      : ( 267.1248, 128.5897 ) dan ( 267.1485, 128.7187 )
Jumlah perhitungan jarak euclidean : 12497500
Waktu perhitungan                   : 18548.004150390625 milidetik

└──────────────────────────────────────────────────────      Brute Force       ──────────────────────────────────────────────────────┘
```

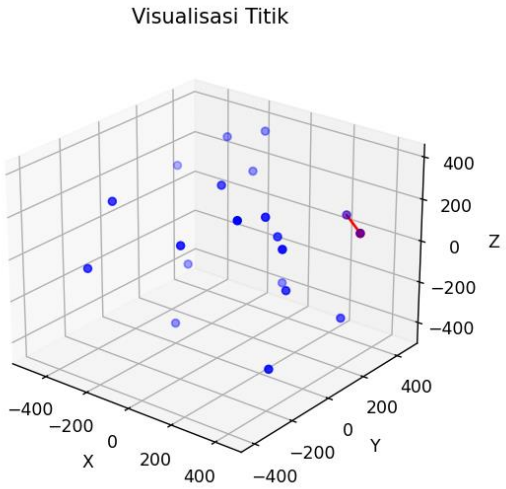


5. Test *handling Input* (*Input* Salah)



6. Test keyboard untuk n = 30 dan d = 3





7. Test keyboard input untuk n = 1000 dan d = 100

```
Masukkan jumlah titik  : 1000
Masukkan dimensi titik : 100
                                        Devide and Conquer

Jarak terdekat adalah              : 2909.5013809352736
Pasangan Titik terdekat adalah     : ( -299.3448, 110.6538, 197.0426, 424.0738, -310.2974, -32.6279, 93.5863, 90.5870, 28.4337, -378.1910, 19.4055, -290.0380, 91.8593, 143.5059, 240.4608, 460.4
380, -305.9705, -299.3087, -115.5780, 170.3118, 353.5914, -179.1281, -284.9746, -7.7677, -348.7453, 431.4592, 212.6951, -330.8593, 117.2845, 477.2311, 256.3297, -395.4644, -264.9291, 62.6596, 2
41.1273, -206.5534, -57.1343, 415.5927, -105.4133, -9.8883, 148.7807, 161.6071, -189.9152, 360.3625, 475.9676, -181.4605, -54.9781, -119.4039, -278.2972, -305.1496, -51.6692, 70.7435, -192.4949
, 181.5282, -213.0056, 91.6586, 158.2858, -463.8017, -138.3492, -231.6521, 177.0169, -205.3197, 75.9113, 221.4614, -235.5508, -72.0798, 279.6003, -406.0560, -228.5198, -275.3897, -162.5826, 167
.4489, 5.8776, -73.2718, 294.1622, -255.8860, 234.8880, 217.8238, 487.9418, -435.2368, -67.6228, -165.0143, 2.3753, -294.9174, -74.5681, 268.8205, -23.2943, -138.1939, 283.6765, 257.7490, -329.
5892, 431.4791, 162.0309, -338.4638, -39.3838, 428.4461, -290.2031, 99.8071, 137.2264, -210.1854 ) dan ( 267.3309, 415.3610, -333.6762, 348.9222, -430.4051, 145.7801, 160.1304, 390.0780, -218.7
545, -453.9003, -295.5720, -363.9192, -353.3594, 424.7617, 41.2737, 330.8456, -274.3943, -97.2030, -53.2244, -3.5910, -176.0931, -145.9405, -71.2458, 141.3199, -87.0676, 455.3952, 67.3789, -133
.7840, -218.5406, 177.6837, 396.9765, 115.3862, -228.9921, 389.7302, 350.2106, 57.3985, 42.5675, 106.0705, -131.1767, 196.8619, 290.9114, 23.2164, 255.6031, -231.0527, -77.7600, 208.5800, -162.
0832, -44.3847, -467.5966, -36.9993, 178.6410, 76.4368, -174.7693, -304.6694, 92.2676, -344.0139, -344.7791, 212.7840, 44.5852, 53.5315, 129.5544, -439.9246, 148.8307, 116.3862, -110.5157, -126
.1283, 243.7153, -435.0509, -4.9601, -16.9898, -236.5191, 264.2480, -69.9226, -480.2398, 102.6885, -90.9152, -129.0133, 153.6674, 369.1546, 479.6153, -323.0488, -439.8728, -423.5458, -177.3536,
 389.9437, 437.7489, -374.9183, -113.7824, -461.3961, 280.1339, -388.6176, 250.5616, 147.3186, -179.5858, -11.6027, 37.2238, -355.2200, -120.3907, -156.0329, 119.4335 )
Jumlah perhitungan jarak euclidean : 499500
Waktu perhitungan                  : 72907.78374671936 milidetik

                                        Devide and Conquer

Tidak dapat melakukan visualisasi
```

```
                                        Brute Force

Jarak terdekat adalah              : 2909.5013809352736
Pasangan Titik terdekat adalah     : ( -299.3448, 110.6538, 197.0426, 424.0738, -310.2974, -32.6279, 93.5863, 90.5870, 28.4337, -378.1910, 19.4055, -290.0380, 91.8593, 143.5059, 240.4608, 460.4
380, -305.9705, -299.3087, -115.5780, 170.3118, 353.5914, -179.1281, -284.9746, -7.7677, -348.7453, 431.4592, 212.6951, -330.8593, 117.2845, 477.2311, 256.3297, -395.4644, -264.9291, 62.6596, 2
41.1273, -206.5534, -57.1343, 415.5927, -105.4133, -9.8883, 148.7807, 161.6071, -189.9152, 360.3625, 475.9676, -181.4605, -54.9781, -119.4039, -278.2972, -305.1496, -51.6692, 70.7435, -192.4949
, 181.5282, -213.0056, 91.6586, 158.2858, -463.8017, -138.3492, -231.6521, 177.0169, -205.3197, 75.9113, 221.4614, -235.5508, -72.0798, 279.6003, -406.0560, -228.5198, -275.3897, -162.5826, 167
.4489, 5.8776, -73.2718, 294.1622, -255.8860, 234.8880, 217.8238, 487.9418, -435.2368, -67.6228, -165.0143, 2.3753, -294.9174, -74.5681, 268.8205, -23.2943, -138.1939, 283.6765, 257.7490, -329.
5892, 431.4791, 162.0309, -338.4638, -39.3838, 428.4461, -290.2031, 99.8071, 137.2264, -210.1854 ) dan ( 267.3309, 415.3610, -333.6762, 348.9222, -430.4051, 145.7801, 160.1304, 390.0780, -218.7
545, -453.9003, -295.5720, -363.9192, -353.3594, 424.7617, 41.2737, 330.8456, -274.3943, -97.2030, -53.2244, -3.5910, -176.0931, -145.9405, -71.2458, 141.3199, -87.0676, 455.3952, 67.3789, -133
.7840, -218.5406, 177.6837, 396.9765, 115.3862, -228.9921, 389.7302, 350.2106, 57.3985, 42.5675, 106.0705, -131.1767, 196.8619, 290.9114, 23.2164, 255.6031, -231.0527, -77.7600, 208.5800, -162.
0832, -44.3847, -467.5966, -36.9993, 178.6410, 76.4368, -174.7693, -304.6694, 92.2676, -344.0139, -344.7791, 212.7840, 44.5852, 53.5315, 129.5544, -439.9246, 148.8307, 116.3862, -110.5157, -126
.1283, 243.7153, -435.0509, -4.9601, -16.9898, -236.5191, 264.2480, -69.9226, -480.2398, 102.6885, -90.9152, -129.0133, 153.6674, 369.1546, 479.6153, -323.0488, -439.8728, -423.5458, -177.3536,
 389.9437, 437.7489, -374.9183, -113.7824, -461.3961, 280.1339, -388.6176, 250.5616, 147.3186, -179.5858, -11.6027, 37.2238, -355.2200, -120.3907, -156.0329, 119.4335 )
Jumlah perhitungan jarak euclidean : 499500
Waktu perhitungan                  : 22523.619413375854 milidetik

                                        Brute Force

Tidak dapat melakukan visualisasi
```

BAB IV
KESIMPULAN DAN SARAN

Saat membuat program penulis menyadari beberapa hal dari algoritma devide and conquer untuk perhitunggan jarak 2 titik terdekat, yaitu :

1. Algoritme tidak selalu efektif, jika setelah membagi terdapat banyak titik yang dekat dengan ’garis khayalan’ akan memerlukan brute force yang lama.
2. Tidak dapat dipungkiri bahwa algoritme ini tetap memerlukan brute force ketika terjadi kejadian seperti poin 1.

LAMPIRAN

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

Repository Github

https://github.com/akmaldika/Tucil2_13521070.git