Laporan Tugas Kecil Strategi Algoritma 2

Pencarian Pasangan Titik Terdekat 3D

dengan Menggunakan Algoritma Divide and Conquer

Disusun oleh:

Nama : Sulthan Dzaky Alfaro

NIM :13521159

Kelas : K1

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

TAHUN 2022/2023

BAB 1

Algoritma Divide and Conquer

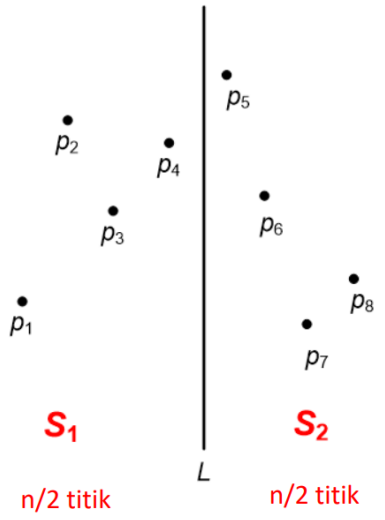
Algoritma Divide and Conquer merupakan algoritma yang dilakukan dengan memecah permasalahan menjadi beberapa bagian kecil agar permasalahan menjadi mudah untuk diselesaikan. Algoritma ini sangatlah efisien, karena memiliki kompleksitas waktu yang lebih kecil dibandingkan beberapa algoritma lain salah satunya brute force. Banyak pengaplikasian algoritma ini. Contohnya seperti pada sorting yaitu merge sort, convex hull, dan salah satunya mencari pasangan titik terdekat.

Pencarian pasangan titik terdekat dapat menggunakan algoritma divide and conquer. Dengan menggunakan prinsip algoritma ini, yaitu memecah-mecah bagian persoalan menjadi kecil dapat digunakan dalam persoalan ini. Dalam tugas kecil strategi algoritma yang ke 2, pencarian pasangan titik diselesaikan dengan menggunakan algoritma divide and conquer. Tahapan pada program ini adalah menerima input, menghasilkan output, dan menampilkan plot pada diagram.

Membaca input

Pada awal program, program akan meminta input berupa banyaknya titik yang akan dites misalkan n titik. Lalu program akan membuat titik secara random sebanyak n buah titik. Lalu program akan mencari pasangan titik dengan menggunakan algoritma divide and conquer. Berikut penjelasan algoritma divide and conquer.

Pertama buat list yang berisi semua titik yang telah dibuat secara random sebelumnya. Lalu sorting/urutkan titik berdasarkan koordinat x. Setelah disorting, bagi list titik titik tersebut menjadi 2 bagian, yaitu bagian kanan dan kiri. Nantinya bagian kanan dan kiri ini akan dibandingkan dan cari yang paling kecil. Berikut contoh pembagian titik titik dalam 2 dimensi



Untuk contoh ini menggunakan 2 dimensi karena sebenarnya penerapan pada 3 dimensi hampir sama dengan yang 2 dimensi. Setelah dibagi menjadi 2, bagi jadi 2 lagi secara

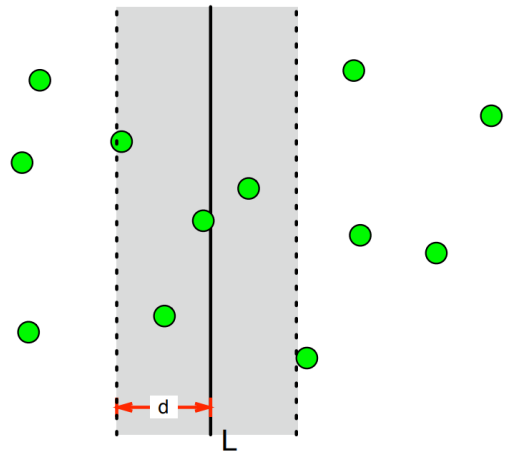
rekursif sampai list menjadi hanya terdiri dari 2 titik atau 3 jika ganjil. Apabila sudah terdiri 2 atau 3 titik, kembalikan hasil jarak antara 2 titik tersebut, jika 3 cari yang terpendek. Untuk mencari jarak digunakan rumus

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

Setelah itu bandingkan jarak tersebut dengan jarak yang tadi telah dibagi, cari yang paling kecil. Begitu seterusnya sampai selesai dan mendapatkan yang terpendek misalkan d. Untuk kemungkinan hasil terpendek, ada 3 kemungkinan yang dapat dihasilkan, yaitu:

- Jarak terpendek terdapat didaerah kanan
- Jarak terpendek terdapat didaerah kiri
- Jarak terpendek terdapat didaerah sekitar pembagi 2

Oleh karena itu kita perlu mengecek daerah disekitar pembagi 2. Yang perlu dicek adalah daerah pada pembagi 2 sebesar d ke kiri dan ke kanan seperti pada contoh gambar berikut



Cek semua titik yang berada didaerah tersebut lalu bandingkan dengan hasil terpendek yang sebelumnya. Setelah semua dilakukan selanjutnya tahap menghasilkan output. Kompleksitas untuk program ini adalah O($nlogn$), untuk *worst case* bisa jadi sampai O(n^2).

Menghasilkan Output

Output yang dihasilkan adalah jarak terpendek pada semua titik titik pada 3 dimensi. Pada program ini, akan menghasilkan 2 jenis output, yaitu jarak terpendek versi algoritma divide and conquer dan versi algoritma brute force. 2 output ini digunakan sebagai pembanding waktu eksekusi sekaligus hasil dari 2 versi tersebut.

Menampilkan Plot pada Diagram

Pada bagian ini, program akan memberikan pilihan, apakah ingin melihat plot titik titik pada diagram bidang 3 dimensi atau tidak. Jika iya, program akan memproses titik titik yang telah dibuat menjadi sebuah diagram scatter plot 3 dimensi. Untuk pasangan titik yang terdekat akan memiliki warna yang berbeda dengan yang lain. Disebelah plot semua titik titik, terdapat plot pasangan titik terdekat agar dapat mengetahui jarak pasangan titik tersebut. Untuk pilihan tidak, program tidak memplotkan titik titik tersebut dan program akan selesai.

BAB 2

Source Code Program

A. Fungsi Penting

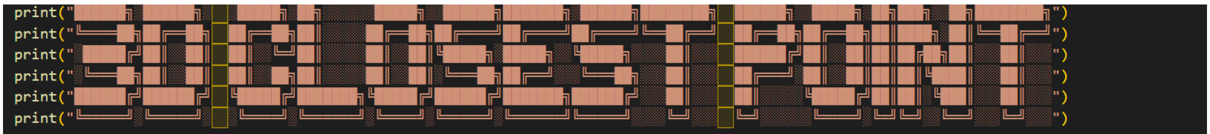
```python
import random
import platform
import time
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d


class Point:
    def __init__(self,x,y,z):
        self.x = x
        self.y = y
        self.z = z
    def __lt__(self,other):
        return self.x < other.x
    def __le__(self,other):
        return self.x <= other.x


def distance(P,Q):
    hasil = ((P.x-Q.x)**2+(P.y-Q.y)**2+(P.z-Q.z)**2)**(1/2)
    return hasil

def mergeSort(x):
    if(len(x)>1):
        tengah = len(x)//2
        kiri = x[:tengah]
        kanan = x[tengah:]
        mergeSort(kanan)
        mergeSort(kiri)
        i = 0
        j = 0
        k = 0
        while(i<len(kiri) and j<len(kanan)):
            if(kiri[i]<=kanan[j]):
                x[k] = kiri[i]
                i+=1
            else:
                x[k] = kanan[j]
                j+=1
            k+=1
        while i<len(kiri):
            x[k] = kiri[i]
            i+=1
            k+=1
        while j<len(kanan):
            x[k] = kanan[j]
            j+=1
            k+=1


def bruteForceDistance(P):
    minim = float("inf")
    count = 0
    start = time.time()
    minimkoor1 = Point(x=0,y=0,z=0)
    minimkoor2 = Point(x=0,y=0,z=0)
    for i in range(len(P)):
```

```
        for j in range(len(P)):
            if(i==j):
                continue
            else:
                count+=1
                hitung = distance(P[i],P[j])
                if(minim>hitung):
                    minim = hitung
                    minimkoor1 = Point(x=P[i].x,y=P[i].y,z=P[i].z)
                    minimkoor2 = Point(x=P[j].x,y=P[j].y,z=P[j].z)
    end = time.time()
    waktu = end-start
    return minim,waktu,minimkoor1,minimkoor2,count

def threepoint(arr):
    min12 = distance(arr[0],arr[1])
    min23 = distance(arr[1],arr[2])
    min13 = distance(arr[0],arr[2])
    minakhir = min(min12,min23,min13)
    return minakhir
def cariTengah(arr,minsmntr):
    batas = 0
    if(len(arr)%2==1):
        batas = arr[len(arr)//2].x
    else:
        batas = (arr[len(arr)//2-1].x + arr[(len(arr)//2)].x)/2
    batasbawah = batas - minsmntr
    batasatas = batas + minsmntr
    termasuk = []
```

```
    for i in range (len(arr)):
        if(arr[i].x>batasbawah and arr[i].x<batasatas):
            termasuk.append(arr[i])
    return termasuk

def minTengah(arrtengah):
    global count
    minim = float("inf")
    for i in range(len(arrtengah)):
        for j in range(i+1,len(arrtengah)):
            count+=1
            jarak2titik = distance(arrtengah[i],arrtengah[j])
            if(minim>jarak2titik):
                minim =  jarak2titik
    return minim
def dividenConquer(arr):
    global count
    if(len(arr)==3):
        return threepoint(arr)
    elif(len(arr)==2):
        count+=1
        return distance(arr[0],arr[1])
    else:
        bagi2 = len(arr)//2
        kiri = arr[:bagi2]
        kanan = arr[bagi2:]
        hasilkanan = dividenConquer(kanan)
        hasilkiri = dividenConquer(kiri)
        minim = min(hasilkanan,hasilkiri)
```

```
        tengaharr = cariTengah(arr,minim)
        hasilmintengah = minTengah(tengaharr)
        hasilakhir = min(minim,hasilmintengah)
        return hasilakhir
```

B. Main Program



```
count = 0
print("Selamat datang di 3D Closest Point")
print("Berikut Spesifikasi Sistem Anda:")
my_system = platform.uname()

print(f"System: {my_system.system}")
print(f"Node Name: {my_system.node}")
print(f"Release: {my_system.release}")
print(f"Version: {my_system.version}")
print(f"Machine: {my_system.machine}")
print(f"Processor: {my_system.processor}")
print("\n")
banyakinput = int(input("Masukkan banyak titik yang akan diuji = "))
while(banyakinput<=1):
    print("Masukan salah, input harus lebih dari 1")
    banyakinput = int(input("Masukkan banyak titik yang akan diuji = "))
listpoint = []
for i in range(banyakinput):
    a = random.randint(-100000,100000)
    b = random.randint(-100000,100000)
    c = random.randint(-100000,100000)
```

```
    koor = Point(x=a,y=b,z=c)
    listpoint.append(koor)
mergeSort(listpoint)
```

```
start = time.time()
hasildnc = dividenConquer(listpoint)
end = time.time()
waktudnc = end-start
print("Jarak terdekat antar titik yang ditemukan : ")
print("---------------Divide and Conquer---------------")
print("Divide and Conquer =",hasildnc)
print("Time Execution =",waktudnc)
print("Banyak operasi =",count)
print("-------------------Brute Force-------------------")
print("Sebentar,lagi ngitung yang bruteforce :D")
hasilbruteForce,waktu,koor1,koor2,oprbrt = bruteForceDistance(listpoint)
print("Brute Force =",hasilbruteForce)
print("Time Execution =",waktu)
print("Banyak operasi =",oprbrt)
print("-------------------------------------------------")
print("Sepasang titik terdekat:")
print("Titik 1 = ("+str(koor1.x)+","+str(koor1.y)+","+str(koor1.z)+")")
print("Titik 2 = ("+str(koor2.x)+","+str(koor2.y)+","+str(koor2.z)+")")


tanya = input("Apakah ingin melihat plot dari titik titik tersebut(y/n)=")
if(tanya=='y' or tanya=='Y'):
    listx = []
    listy = []
    listz = []

    for i in range(len(listpoint)):
        if((listpoint[i].x==koor1.x and listpoint[i].y==koor1.y and listpoint[i].z==koor1.z) or (listpoint[i].x==koor2.x and listpoint[i].y
```

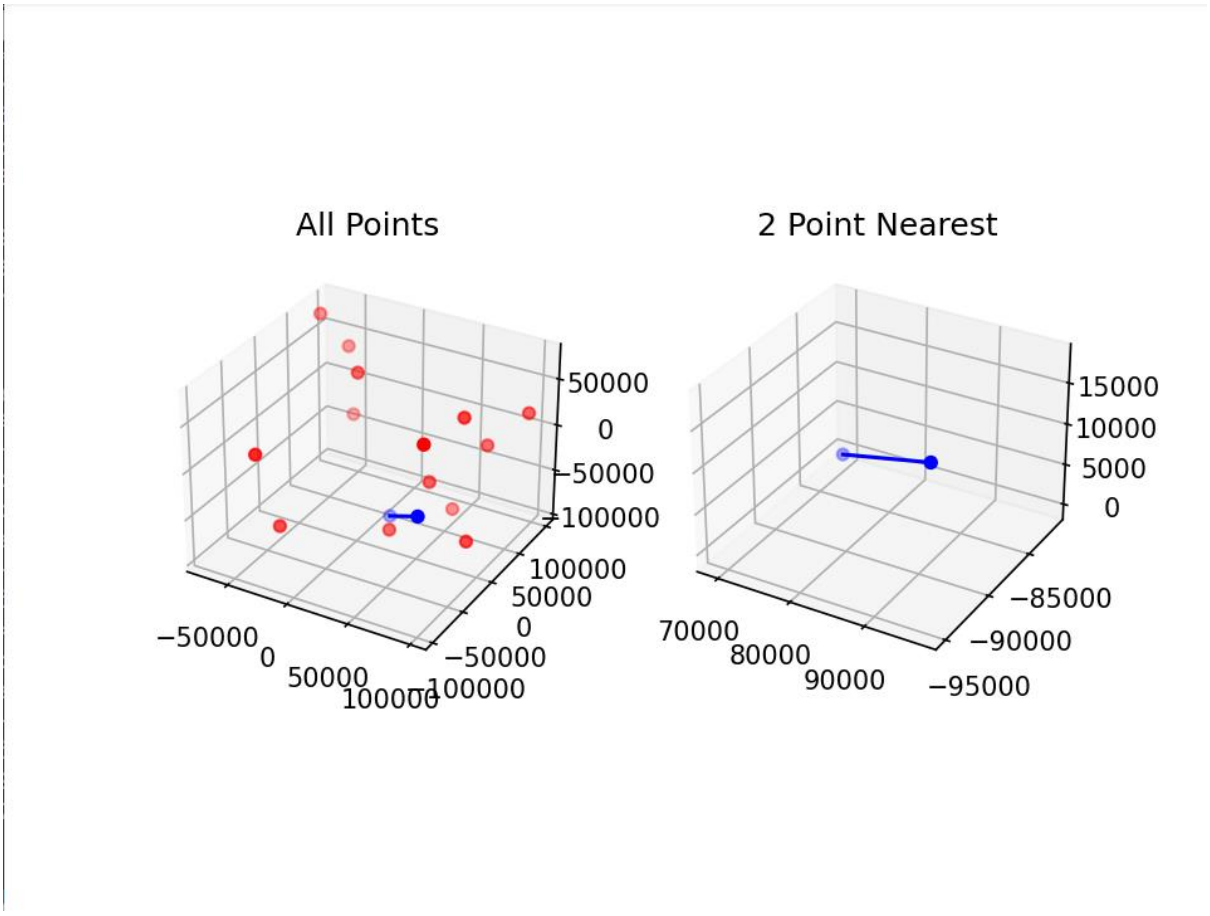
```
            continue
        else:
            listx.append(listpoint[i].x)
            listy.append(listpoint[i].y)
            listz.append(listpoint[i].z)

    numx = np.array(listx)
    numy = np.array(listy)
    numz = np.array(listz)

    hasilkoorx = np.array([koor1.x,koor2.x])
    hasilkoory = np.array([koor1.y,koor2.y])
    hasilkoorz = np.array([koor1.z,koor2.z])

    fig = plt.figure()

    project  =fig.add_subplot(121,projection = "3d")
    project.set_title("All Points")
    project.scatter(hasilkoorx,hasilkoory,hasilkoorz,c = "b")
    project.scatter(numx,numy,numz,c = "r")
    project.plot(hasilkoorx,hasilkoory,hasilkoorz,c = "b")

    project2  =fig.add_subplot(122,projection = "3d")
    project2.set_title("2 Point Nearest")
    project2.scatter(hasilkoorx,hasilkoory,hasilkoorz,c = "b")
    project2.plot(hasilkoorx,hasilkoory,hasilkoorz,c = "b")

    plt.show()
else:
    print("Ok, titik titik tidak diplotkan. Program telah berakhir")
```

BAB 3

Screenshot Contoh Input dan Output

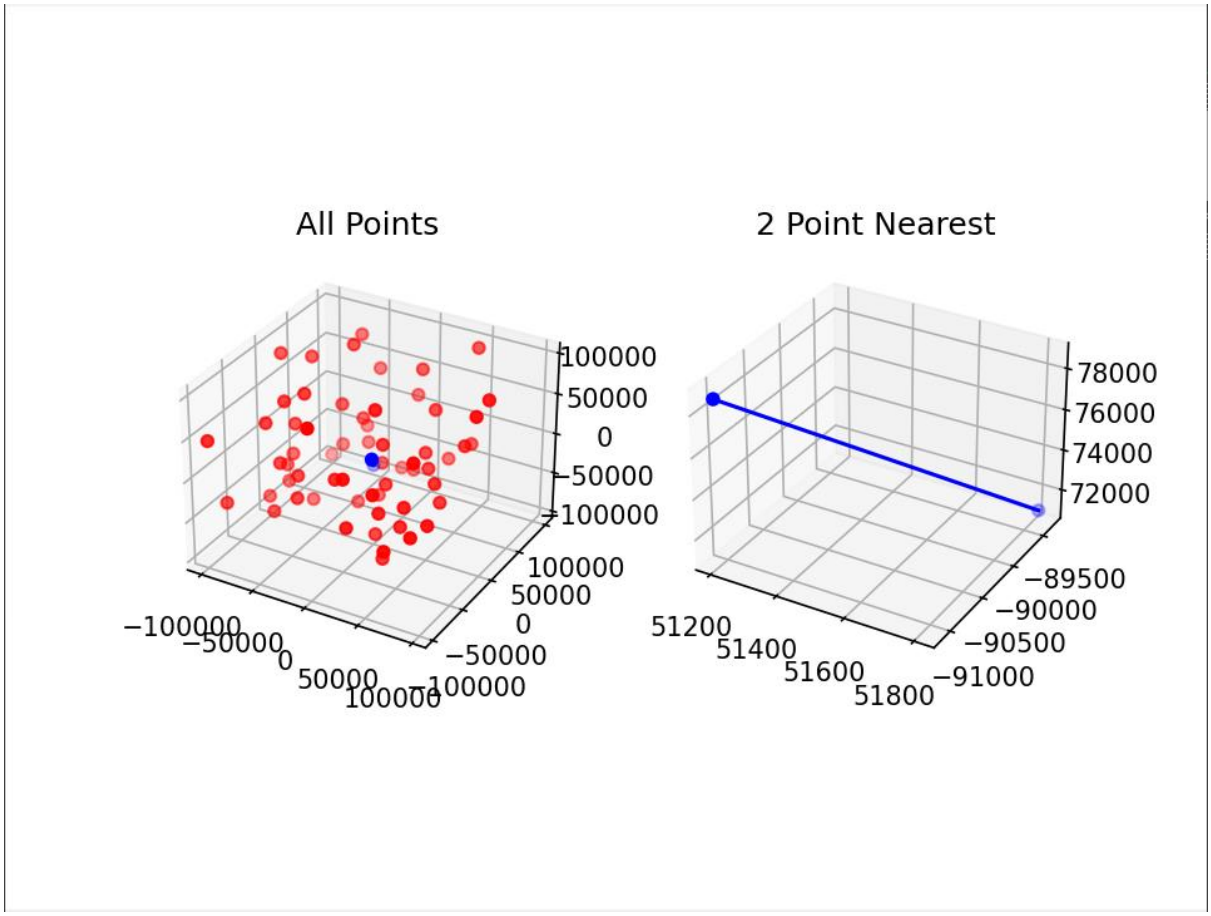
1. n = 16

```
3D CLOSEST POINT
Selamat datang di 3D Closest Point
Berikut Spesifikasi Sistem Anda:
System: Windows
Node Name: DESKTOP-5V3BCPU
Release: 10
Version: 10.0.22621
Machine: AMD64
Processor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel


Masukkan banyak titik yang akan diuji = 16
Jarak terdekat antar titik yang ditemukan :
----------Divide and Conquer----------
Divide and Conquer = 36479.93418305466
Time Execution = 0.0
Banyak operasi = 90
----------Brute Force----------
Sebentar,lagi ngitung yang bruteforce :D
Brute Force = 36479.93418305466
Time Execution = 0.0
Banyak operasi = 240
--------------------------------------------------
Sepasang titik terdekat:
Titik 1 = (69107,-81615,-595)
Titik 2 = (97270,-95085,18278)
```



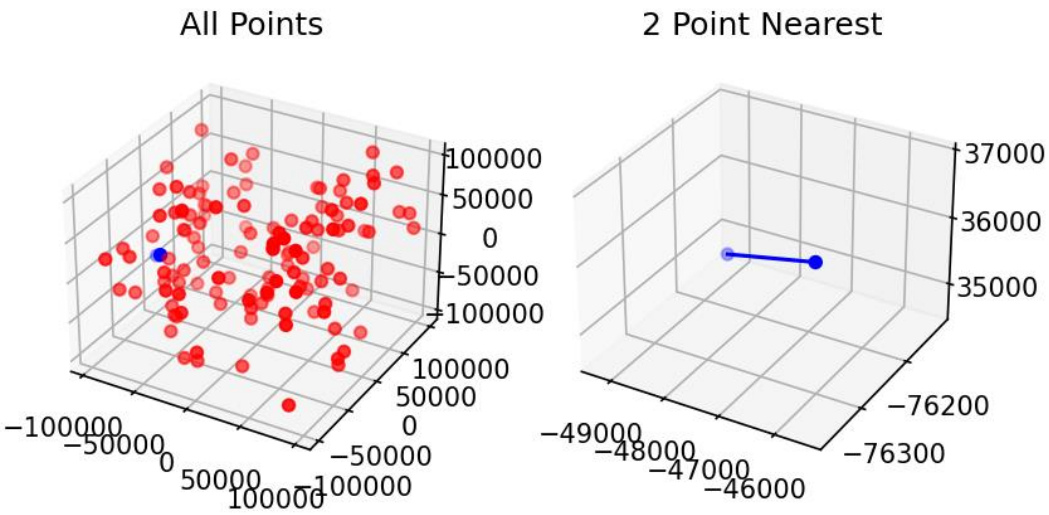
2. n = 64

```
3D CLOSEST POINT
Selamat datang di 3D Closest Point
Berikut Spesifikasi Sistem Anda:
System: Windows
Node Name: DESKTOP-5V3BCPU
Release: 10
Version: 10.0.22621
Machine: AMD64
Processor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel


Masukkan banyak titik yang akan diuji = 64
Jarak terdekat antar titik yang ditemukan :
----------Divide and Conquer----------
Divide and Conquer = 7810.378351911001
Time Execution = 0.007697343826293945
Banyak operasi = 562
----------Brute Force----------
Sebentar,lagi ngitung yang bruteforce :D
Brute Force = 7810.378351911001
Time Execution = 0.0
Banyak operasi = 4032
--------------------------------------------------
Sepasang titik terdekat:
Titik 1 = (51200,-91111,78678)
Titik 2 = (51812,-89326,71099)
```



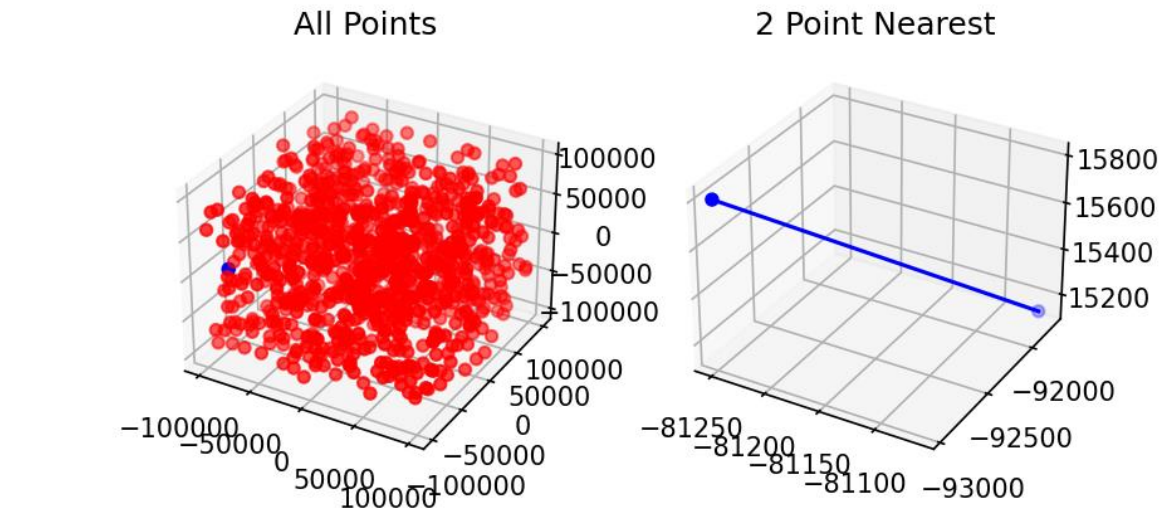
3. n = 128

```
3D CLOSEST POINT
Selamat datang di 3D Closest Point
Berikut Spesifikasi Sistem Anda:
System: Windows
Node Name: DESKTOP-5V3BCPU
Release: 10
Version: 10.0.22621
Machine: AMD64
Processor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel


Masukkan banyak titik yang akan diuji = 128
Jarak terdekat antar titik yang ditemukan :
---------------Divide and Conquer----------------
Divide and Conquer = 4456.749712514716
Time Execution = 0.0
Banyak operasi = 1599
------------------Brute Force--------------------
Sebentar,lagi ngitung yang bruteforce :D
Brute Force = 4456.749712514716
Time Execution = 0.016809701919555664
Banyak operasi = 16256
-------------------------------------------------
Sepasang titik terdekat:
Titik 1 = (-49278,-76122,34617)
Titik 2 = (-45469,-76366,36918)
```



4. n = 1000

```
3D CLOSEST POINT
Selamat datang di 3D Closest Point
Berikut Spesifikasi Sistem Anda:
System: Windows
Node Name: DESKTOP-5V3BCPU
Release: 10
Version: 10.0.22621
Machine: AMD64
Processor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel


Masukkan banyak titik yang akan diuji = 1000
Jarak terdekat antar titik yang ditemukan :
----------Divide and Conquer----------
Divide and Conquer = 1401.9258182942492
Time Execution = 0.02339005470275879
Banyak operasi = 42080
----------Brute Force----------
Sebentar,lagi ngitung yang bruteforce :D
Brute Force = 1401.9258182942492
Time Execution = 0.43024778366088867
Banyak operasi = 999000
--------------------------------------------------
Sepasang titik terdekat:
Titik 1 = (-81252,-92975,15803)
Titik 2 = (-81062,-91755,15139)
```



Tambahan

Link Github : https://github.com/SulthanDA28/Tucil2_13521159.git

No	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		