# LAPORAN TUGAS KECIL 2
# IF2211 STRATEGI ALGORITMA
# "MENCARI PASANGAN TITIK TERDEKAT 3D DENGAN ALGORITMA *DIVIDE AND CONQUER*"


Disusun oleh:
Moh. Aghna Maysan Abyan (13521076)
Ulung Adi Putra (13521122)

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

SEMESTER II TAHUN AJARAN 2022/2023

# DAFTAR ISI

# BAB I
# DESKRIPSI MASALAH

Tucil 2 meminta mahasiswa untuk 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:

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

Buatlah program dalam dalam Bahasa C/C++/Java/Python/Golang/Ruby/Perl (pilih salah satu) untuk mencari sepasang titik yang jaraknya terdekat datu sama lain dengan menerapkan algoritma divide and conquer untuk penyelesaiannya, dan perbandingannya dengan Algoritma Brute Force.

Masukan program:

- n
- itik-titik (dibangkitkan secara acak) dalam koordinat (x, y, z)

Luaran program:

- sepasang titik yang jaraknya terdekat dan nilai jaraknya
- banyaknya operasi perhitungan rumus Euclidian
- waktu riil dalam detik (spesifikasikan komputer yang digunakan)

Bonus

**Bonus 1** (Nilai = 7,5): Penggambaran semua titik dalam bidang 3D, sepasang titik yang jaraknya terdekat ditunjukkan dengan warna yang berbeda dari titik lainnya.

**Bonus 2** (nilai = 7,5): Generalisasi program anda sehingga dapat mencari sepasang titik terdekat untuk sekumpulan vektor di Rn, setiap vektor dinyatakan dalam bentuk x = (x1, x2, …, xn)

# BAB II
# ALGORITMA *DIVIDE AND CONQUER*

## 2.1 Tentang Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* terdiri dari dua kata, yaitu *divide* dan *conquer*. *Divide* berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama), sedangkan *Conquer (solve)* berarti menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Jika digabungkan, maka algoritma *Divide and Conquer* adalah algoritma yang mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Contoh-contoh persoalan yang diselesaikan dengan algoritma *Divide and Conquer* adalah sebagai berikut:

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

## 2.2 Penjelasan Algoritma *Divide and Conquer*

- `createRandomPoint`

  Fungsi ini akan menerima parameter n yang merepresentasikan banyak titik dan dimensi yang merepresentasikan dimensi dari titik. Program akan mengeluarkan *array* yang berisi point point berdimensi sesuasi masukan dengan jumlah n.

- `euclidianDistance`

  Fungsi ini akan menerima dua buah *point* sebagai parameter yang kemudian akan mengembalikan jarak kedua titik tersebut.

- `Visualize`

  Fungsi ini akan memvisualisasikan semua titik dari *arrayOfPoints*. Pasangan titik dengan jarak terpendek akan diberi warna merah dan dihubungkan dengan garis merah. Titik lainnya akan diberi warna biru.

- `BFSolution`

  Fungsi ini akan menerima *arrayOfPoints* sebagai parameter, dan kemudian akan mengeluarkan jarak terdekat dari kedua titik pada array tersebut beserta koordinat titik, dan banyak operasi `euclidianDistance` yang dilakukan.

- `quickSort`

  Fungsi ini akan menerima *arrayOfPoints* sebagai parameter dan kemudian akan mengembalikan *array* tersebut dengan tiap *point*-nya sudah terurut membesar berdasarkan nilai komponen X. Cara kerja dari fungsi ini adalah menggunakan algoritma *Divide and Conquer*. Jika *array* adalah *array* 0 atau berukuran 1, maka fungsi akan langsung mengembalikan *array* tersebut. Jika tidak, maka *arrayOfPoints* akan dibagai menjadi 3 buah *array* yaitu "left", "equal", dan "right".

  Pertama akan dipilih elemen tengah *arrayOfPoints* sebagai pivot. Lakukan iterasi pada semua titik pada array, jika nilai x pada titik lebih kecil dari nilai x pada pivot, maka titik akan dimasukkan kedalam array "left". Jika nilai x pada titik sama dengan nilai x pada pivot maka titik akan dimasukkan kedalam array "equal". Jika nilai x pada titik lebih besar pada nilai x pada pivot maka titik akan dimasukkan kedalam *array* "right". Ketika sudah dibagi (*divide*) maka *array* "left" dan "right" akan dilakukan proses seperti tadi hingga *array* hasil *divide* berukuran 1. Kemudian hasil tersebut akan digabungkan (*combine*).

- `closestPairWithDnC`

  Fungsi ini akan menerima *arrayOfPoints* yang sudah terurut berdasarkan X sebagai parameter dan akan mengembalikan jarak terpendek antara 2 titik pada *array* tersebut, pasangan titik dengan jarak terpendek, dan banyaknya operasi `euclidianDistance` yang dilakukan. Fungsi ini dikerjakan dengan algoritma *Divide and Conquer*. Jika ukuran *arrayOfPoints* lebih kecil atau sama dengan 3, maka penyelesaian akan langsung dikerjakan menggunakan algoritma *brute-force*. Namun jika tidak, *array* akan dibagi menjadi 2 bagian (*divide*), bagian pertama adalah "left", yaitu *array* yang berisi elemen elemen *arrayOfPoints* dari indeks ke 0 hingga indeks tengah *arrayOfPoints*, bagian "right" berisi elemen-elemen *arrayOfPoints* dari indeks tengah hingga indeks terakhir.

  Kemudian akan dicari jarak terpendek dari "left" dan jarak terpendek dari "right". Jarak terpendek dari "left" dan "right" akan dibandingkan, hasil yang lebih kecil akan diambil sebagai jarak terpendek *arrayOfPoints* sementara. Kemudian akan diambil nilai koordinat X yang memisahkan *arrayOfPoints* menjadi "left" dan "right". Kemudian dicari semua titik yang koordinat X nya berada pada rentang (*middle* - jarak terpendek sementara) hingga (*middle* + jarak terpendek sementara). Titik-titik pada range tersebut akan dimasukkan kedalam *array* "inMiddleRange". Kemudian akan dicari jarak terpendek pada *array* "inMiddleRange" menggunakan algoritma *brute-force*. Namun

untuk meminimalisir perhitungan, pasangan titik yang jarak berdasarkan koordinat Y nya lebih besar dari jarak sementara tidak akan dimasukkan dalam perhitungan. Jarak terpendek dari *keseluruhan arrayOfPoints* akan ditentukan dengan mengambil nilai yang lebih kecil antara jarak terpendek sementara dan hasil *brute-force* pada *array* “inMiddleRange.”

# BAB III
# KODE PROGRAM

## 3.1 Utility.py

```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import random

#point = array dengan ukuran dimensi (default 3 dimensi)
#array of point array yang berisi point point
def createRandomPoint (n, dimensy = 3):
    arrayOfPoint = []
    for i in range(n):
        Point = []
        for j in range(dimensy):
            temp = random.uniform(-1e6,1e6)
            Point.append(temp)
        arrayOfPoint.append(Point)
    return arrayOfPoint

def euclidianDistance (Point1, Point2):
    disctance = 0
    for i in range(len(Point1)):
        disctance += (Point2[i] - Point1[i])**2
    disctance = disctance**(1/2)
    return disctance
```

```python

    def BFSolution (arrayOfPoint):
        shortest = 9999999999999
        Point1 = []
        Point2 = []
        count = 0
        for i in range(0, len(arrayOfPoint) - 1):
            for j in range(i+1, len(arrayOfPoint)):
                distance = euclidianDistance(arrayOfPoint[i], arrayOfPoint[j])
                count += 1
                if(distance < shortest):
                    shortest = distance
                    Point1 = arrayOfPoint[i]
                    Point2 = arrayOfPoint[j]

        return shortest, Point1, Point2, count

```

```python


def quicksort(arrayOfPoints):
    if(len(arrayOfPoints) == 0):
        return []
    elif (len(arrayOfPoints) == 1):
        return arrayOfPoints
    else:
        pivot = arrayOfPoints[len(arrayOfPoints) // 2][0] # Select the middle element as pivot based on x-value
        left = []
        equal = []
        right = []

        for point in arrayOfPoints:
            if point[0] < pivot:
                left.append(point)
            elif point[0] == pivot:
                equal.append(point)
            else:
                right.append(point)

        return quicksort(left) + equal + quicksort(right)

```

```python
def visualize (arrayOfPoint, Point1, Point2):
    if(len(Point1) > 3):
        print("Gabisa divisualisaiin gan, kamu bukan dewa yang bisa liat 3 dimensi keatas !!!!!")
    elif(len(Point1) == 1):
        print("Satu dimensi gabisa divisualisasiin gan! ")
    else:
        if(len(Point1) == 3):
            fig = plt.figure()
            ax = fig.add_subplot(111, projection='3d')

            #X,Y,Z adalah list yang digunakan untuk menampung komponen x,y,z dari point
            X = []
            Y = []
            Z = []
            for point in (arrayOfPoint):
                if(point != Point2 and point != Point1):
                    #jika point berbeda dari 2 point terdekat, maka nilai komponen x,y,z akan dimasukan kedalam list
                    X.append(point[0])
                    Y.append(point[1])
                    Z.append(point[2])

            ax.scatter(X, Y, Z, color = "blue") #warnai biru untuk point yang bukan dua point terdekat

            #untuk 2 point terdekat akan diberi warna merah
            ax.scatter(Point1[0], Point1[1], Point1[2], color = "red")
            ax.scatter(Point2[0], Point2[1], Point2[2], color = "red")
            #memberi garis warna merah yang menghubungkan dua titik terdekat
            ax.plot([Point1[0], Point2[0]], [Point1[1], Point2[1]], [Point1[2], Point2[2]], color = "red")

            # Set the axis labels
            ax.set_xlabel('X')
            ax.set_ylabel('Y')
            ax.set_zlabel('Z')
```

```python
            plt.show()
        elif(len(Point1) == 2):
            X = []
            Y = []
            for point in (arrayOfPoint):
                if(point != Point2 and point != Point1):
                    #jika point berbeda dari 2 point terdekat, maka nilai komponen x,y,z akan dimasukan kedalam list
                    X.append(point[0])
                    Y.append(point[1])

            plt.scatter(X,Y, color = "blue")
            plt.scatter(Point1[0], Point1[1], color = "red")
            plt.scatter(Point2[0], Point2[1], color = "red")
            plt.scatter([Point1[0], Point2[0]], [Point1[1], Point2[1]], color = "red")
            plt.xlabel('X-axis')
            plt.ylabel('Y-axis')

            plt.show()
```

```python
def closestPairWithDnC (arrayOfPoint):
    if (len(arrayOfPoint) <= 3):
        #jika ukuran array kurang atau sama dengan 3, maka akan langsung diselesaikan menggunakan bruteforce
        return BFSolution(arrayOfPoint)
    else:
        mid = len(arrayOfPoint)//2
        #split array menjadi 2 bagian
        left = arrayOfPoint[:mid]
        right = arrayOfPoint[mid:]
        middleX = (left[len(left)-1][0] + right[0][0])/2 #koordinat X yang membagi arrayOfPoints menjadi 2
        #divide
        distanceLeft , Point1L, Point2L, count1 = closestPairWithDnC(left)
        distanceRight, Point1R, Point2R, count2 = closestPairWithDnC(right)
        count = count1 + count2 #banyak operasi perhitungan
        closest = 0
        Point1 = []
        Point2 = []
        #conquer
        if(distanceLeft > distanceRight):
            closest = distanceRight
            Point1 = Point1R
            Point2 = Point2R
        else:
            closest = distanceLeft
            Point1 = Point1L
            Point2 = Point2L
```

```python
    inMiddleRange = []
    for i in range(0, len(arrayOfPoint)):
        if(abs(arrayOfPoint[i][0]- middleX) < closest):
            #mencari titik titik yang berada di range middle - closest sampai middle + closest
            inMiddleRange.append(arrayOfPoint[i])
    for i in range (0, len(inMiddleRange)-1):
        for j in range(i+1, len(inMiddleRange)):
            if(abs(inMiddleRange[j][1] - inMiddleRange[i][1]) < closest):
                distanceMid = euclidianDistance(inMiddleRange[j], inMiddleRange[i])
                count +=1
                if(distanceMid < closest):
                    closest = distanceMid
                    Point1 = inMiddleRange[i]
                    Point2 = inMiddleRange[j]

    return closest, Point1, Point2, count
```

## 3.2 main.py

```python
import Utility as util
import time
from colorama import Fore, Back, Style

print(Fore.RED + "
print(Fore.YELLOW + "
print(Fore.YELLOW + "
print(Fore.GREEN + "
print(Fore.BLUE + "
print(Fore.BLUE + "
print(Style.RESET_ALL)

dimension = int(input("Mau berapa dimensi gan? "))
while(dimension <= 1):
    print("minimal 2 dimensi woy!! ulangi input!! ")
    dimension = int(input("Mau berapa dimensi gan? "))

nPoint = int(input("Mau berapa titik gan? "))
while(nPoint <0):
    print("Yakali jumlah titik negatif! ulangi input!! ")
    nPoint = int(input("Mau berapa titik gan? "))

arrayOfPoints = util.createRandomPoint(nPoint, dimension)
arrayOfPoints = util.quicksort(arrayOfPoints)
startBF = time.time()
shortest, Point1, Point2, count1 = util.BFSolution(arrayOfPoints)
endBF = time.time()

BFtime = endBF - startBF


startDnC = time.time()
shortestdnc, Point1dnc, Point2dnc, count2= util.closestPairWithDnC(arrayOfPoints)
endDnC = time.time()
DnCtime = endDnC - startDnC
```

```
print(Fore.CYAN)
print("====================================Brute Force Solution====================================")
print("Jarak Terdekat                      : " +str(shortest))
print("Koordinat titik pertama             : " +str(Point1))
print("Koordinat titik kedua               : " +str(Point2))
print("Banyak operasi euclidian distance   : " +str(count1) + " total operasi ")
print("Waktu eksekusi                      : " +str(BFtime) + " detik (spek = Intel Core i7)")

print(Fore.MAGENTA)
print("====================================Divide and Conquer Solution====================================")
print("Jarak Terdekat                      : " +str(shortestdnc))
print("Koordinat titik pertama             : " +str(Point1dnc))
print("Koordinat titik kedua               : " +str(Point2dnc))
print("Banyak operasi euclidian distance   : " +str(count2) + " total operasi ")
print("Waktu eksekusi                      : " +str(DnCtime) + " detik (spek = Intel Core i7)")

print(Style.RESET_ALL)
visualise = input("Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! ")

if(visualise == "Y" or visualise == "y"):
    util.visualize(arrayOfPoints, Point1dnc, Point2dnc)
    print("Bye!!")
else:
    print("Bye!!")
```

# BAB IV
# CONTOH MASUKAN DAN LUARAN

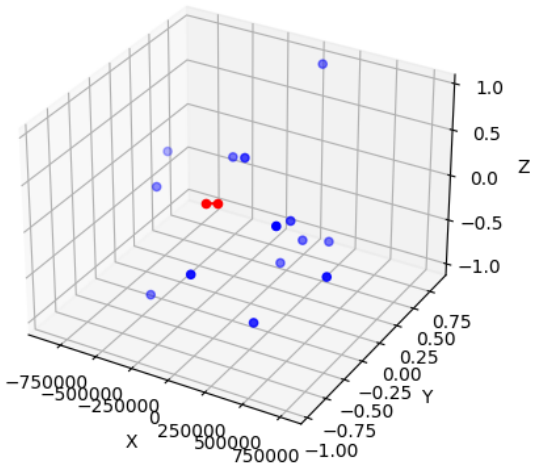
## 4.1 Dimensi 3

### 4.1.1 n = 16

```
PS C:\Users\MSI\OneDrive\Dokumen\Kuliah\Kuliah Semester 4\Strategi Algoritma\Tucil2_13521076_13521122\Tucil2_13521076_
13521122> & C:/Users/MSI/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/MSI/OneDrive/Dokumen/Kuliah/Kuli
ah Semester 4/Strategi Algoritma/Tucil2_13521076_13521122/Tucil2_13521076_13521122/src/main.py"
```

```
 ____  _____ _        _    __  __    _  _____    ____    _  _____  _    _   _  ____
/ ___|| ____| |      / \  |  \/  |  / \|_   _|  |  _ \  / \|_   _|/ \  | \ | |/ ___|
\___ \|  _| | |     / _ \ | |\/| | / _ \ | |    | | | |/ _ \ | | / _ \ |  \| | |  _
 ___) | |___| |___ / ___ \| |  | |/ ___ \| |    | |_| / ___ \| |/ ___ \| |\  | |_| |
|____/|_____|_____/_/   \_\_|  |_/_/   \_\_|    |____/_/   \_\_/_/   \_\_| \_|\____|
```

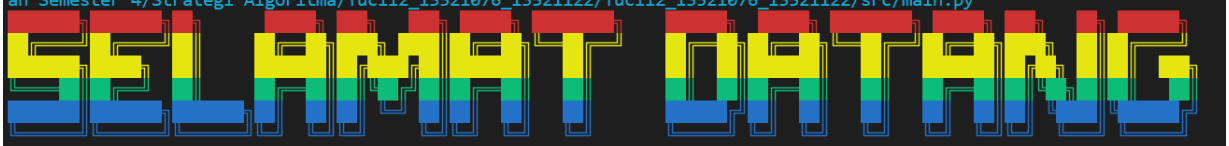
```
Mau berapa dimensi gan? 3
Mau berapa titik gan? 16

=====================================Brute Force Solution=====================================
Jarak Terdekat                      : 84715.62116778079
Koordinat titik pertama             : [-148685.68217593373, -306329.0630298365, 142710.15321538947]
Koordinat titik kedua               : [-105452.83535194444, -239904.08625194523, 112787.25420154002]
Banyak operasi euclidian distance   : 120 total operasi (spek = Intel Core i7)
Waktu eksekusi                      : 0.0 detik

=====================================Divide and Conquer Solution=====================================
Jarak Terdekat                      : 84715.62116778079
Koordinat titik pertama             : [-148685.68217593373, -306329.0630298365, 142710.15321538947]
Koordinat titik kedua               : [-105452.83535194444, -239904.08625194523, 112787.25420154002]
Banyak operasi euclidian distance   : 24 total operasi (spek = Intel Core i7)
Waktu eksekusi                      : 0.0 detik

Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!!
```



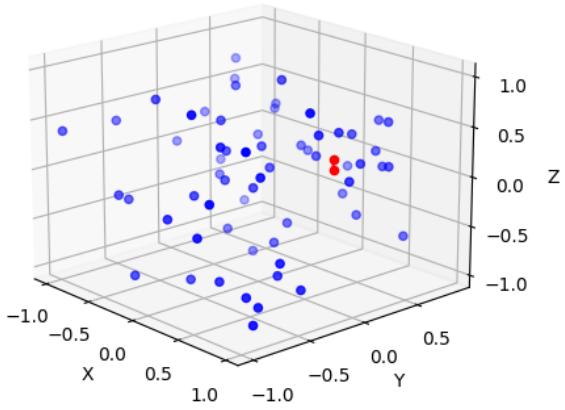

```
Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! y
Bye!!
```

### 4.1.2 n = 64

```
PS C:\Users\MSI\OneDrive\Dokumen\Kuliah\Kuliah Semester 4\Strategi Algoritma\Tucil2_13521076_13521122\Tucil2_13521076_
13521122> & C:/Users/MSI/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/MSI/OneDrive/Dokumen/Kuliah/Kuli
ah Semester 4/Strategi Algoritma/Tucil2_13521076_13521122/Tucil2_13521076_13521122/src/main.py"
```




```
Mau berapa dimensi gan? 3
Mau berapa titik gan? 64

====================================Brute Force Solution====================================
Jarak Terdekat                          : 96967.85083017174
Koordinat titik pertama                 : [62070.95347952307, 654023.0165942379, -46831.35505569202]
Koordinat titik kedua                   : [89208.59753132379, 632993.3416193174, -137517.98659744358]
Banyak operasi euclidian distance       : 2016 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.003005504608154297 detik

==================================Divide and Conquer Solution==================================
Jarak Terdekat                          : 96967.85083017174
Koordinat titik pertama                 : [62070.95347952307, 654023.0165942379, -46831.35505569202]
Koordinat titik kedua                   : [89208.59753132379, 632993.3416193174, -137517.98659744358]
Banyak operasi euclidian distance       : 163 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.0009999275207519531 detik

Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!!
```

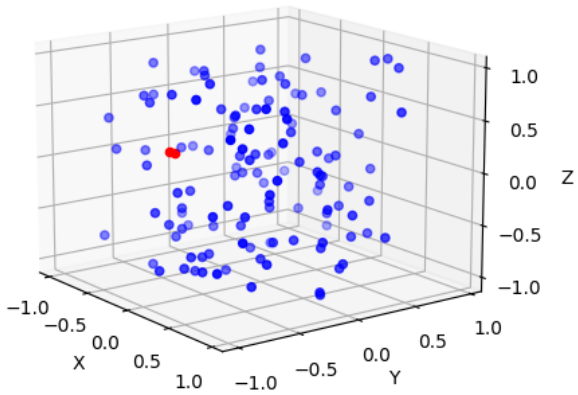



```
Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! y
Bye!!
```

### 4.1.3 n = 128

```
PS C:\Users\MSI\OneDrive\Dokumen\Kuliah\Kuliah Semester 4\Strategi Algoritma\Tucil2_13521076_13521122\Tucil2_13521076_
13521122> & C:/Users/MSI/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/MSI/OneDrive/Dokumen/Kuliah/Kuli
ah Semester 4/Strategi Algoritma/Tucil2_13521076_13521122/Tucil2_13521076_13521122/src/main.py"
```

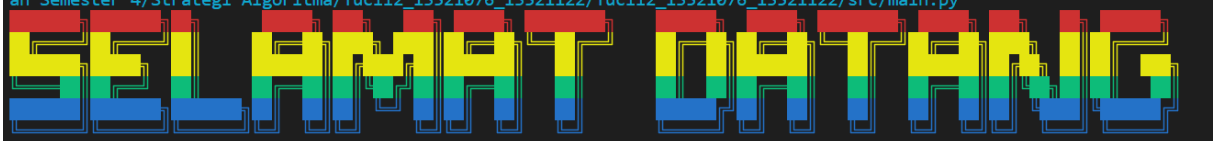

```
Mau berapa dimensi gan? 3
Mau berapa titik gan? 128

====================================Brute Force Solution====================================
Jarak Terdekat                          : 44065.170055251474
Koordinat titik pertama                 : [-508355.1456610134, -426775.1878478456, 73802.00506357709]
Koordinat titik kedua                   : [-498526.7868374341, -386240.9072978947, 59585.30097539001]
Banyak operasi euclidian distance       : 8128 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.01062631607055664 detik

====================================Divide and Conquer Solution====================================
Jarak Terdekat                          : 44065.170055251474
Koordinat titik pertama                 : [-508355.1456610134, -426775.1878478456, 73802.00506357709]
Koordinat titik kedua                   : [-498526.7868374341, -386240.9072978947, 59585.30097539001]
Banyak operasi euclidian distance       : 412 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.0020275115966796875 detik

Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!!
```



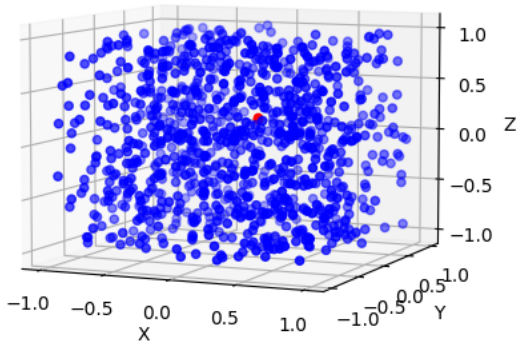

```
Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! y
Bye!!
```

### 4.1.4 n = 1000

```
PS C:\Users\MSI\OneDrive\Dokumen\Kuliah\Kuliah Semester 4\Strategi Algoritma\Tucil2_13521076_13521122\Tucil2_13521076_
13521122> & C:/Users/MSI/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/MSI/OneDrive/Dokumen/Kuliah/Kuli
ah Semester 4/Strategi Algoritma/Tucil2_13521076_13521122/Tucil2_13521076_13521122/src/main.py"
```




```
Mau berapa dimensi gan? 3
Mau berapa titik gan? 1000

====================================Brute Force Solution====================================
Jarak Terdekat                          : 3340.8955246660166
Koordinat titik pertama                 : [401020.20481875725, -539631.2918991682, 284347.0217854453]
Koordinat titik kedua                   : [403736.523908657, -539069.6162707922, 282484.8418808528]
Banyak operasi euclidian distance       : 499500 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.460796594619751 detik

==================================Divide and Conquer Solution==================================
Jarak Terdekat                          : 3340.8955246660166
Koordinat titik pertama                 : [401020.20481875725, -539631.2918991682, 284347.0217854453]
Koordinat titik kedua                   : [403736.523908657, -539069.6162707922, 282484.8418808528]
Banyak operasi euclidian distance       : 4514 total operasi (spek = Intel Core i7)
Waktu eksekusi                          : 0.014229774475097656 detik

Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!!
```




```
Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! y
Bye!!
```

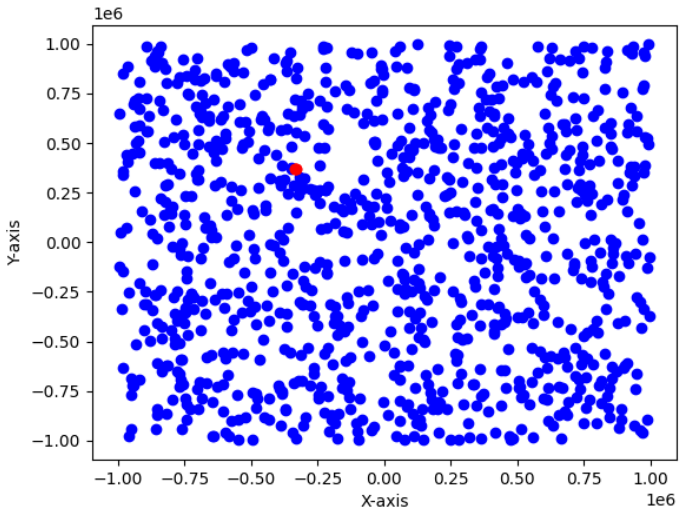
## 4.2 Dimensi selain 3

### 4.2.1 Dimensi 2, n = 1000

```
& C:/Users/MSI/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/MSI/OneDrive/Dokumen/Kuliah/Kuliah Semester 4/Strategi Algoritma/Tucil2_13521076_13521122/Tucil2_13521076_13521122/src/main.py"
```

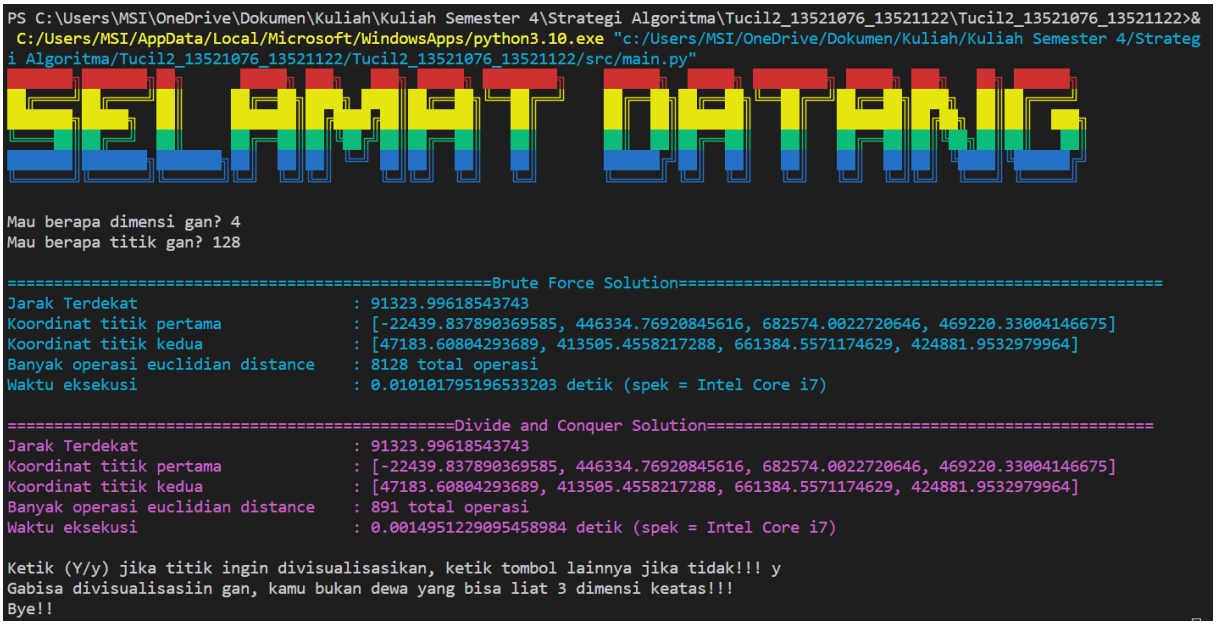
```
 ____  _____ _        _    __  __    _  _____   ____    _  _____  _    _   _  ____
/ ___|| ____| |      / \  |  \/  |  / \|_   _| |  _ \  / \|_   _|/ \  | \ | |/ ___|
\___ \|  _| | |     / _ \ | |\/| | / _ \ | |   | | | |/ _ \ | | / _ \ |  \| | |  _
 ___) | |___| |___ / ___ \| |  | |/ ___ \| |   | |_| / ___ \| |/ ___ \| |\  | |_| |
|____/|_____|_____/_/   \_\_|  |_/_/   \_\_|   |____/_/   \_\_/_/   \_\_| \_|\____|
```

```
Mau berapa dimensi gan? 2
Mau berapa titik gan? 1000

==================================================Brute Force Solution==================================================
Jarak Terdekat                          : 1880.0054662925404
Koordinat titik pertama                 : [-334062.260261639, 369874.3965775608]
Koordinat titik kedua                   : [-332685.4958130317, 371154.60757827153]
Banyak operasi euclidian distance       : 499500 total operasi
Waktu eksekusi                          : 0.3752613067626953 detik (spek = Intel Core i7)

================================================Divide and Conquer Solution=================================================
Jarak Terdekat                          : 1880.0054662925404
Koordinat titik pertama                 : [-334062.260261639, 369874.3965775608]
Koordinat titik kedua                   : [-332685.4958130317, 371154.60757827153]
Banyak operasi euclidian distance       : 1206 total operasi
Waktu eksekusi                          : 0.004755258560180664 detik (spek = Intel Core i7)

Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!!
```

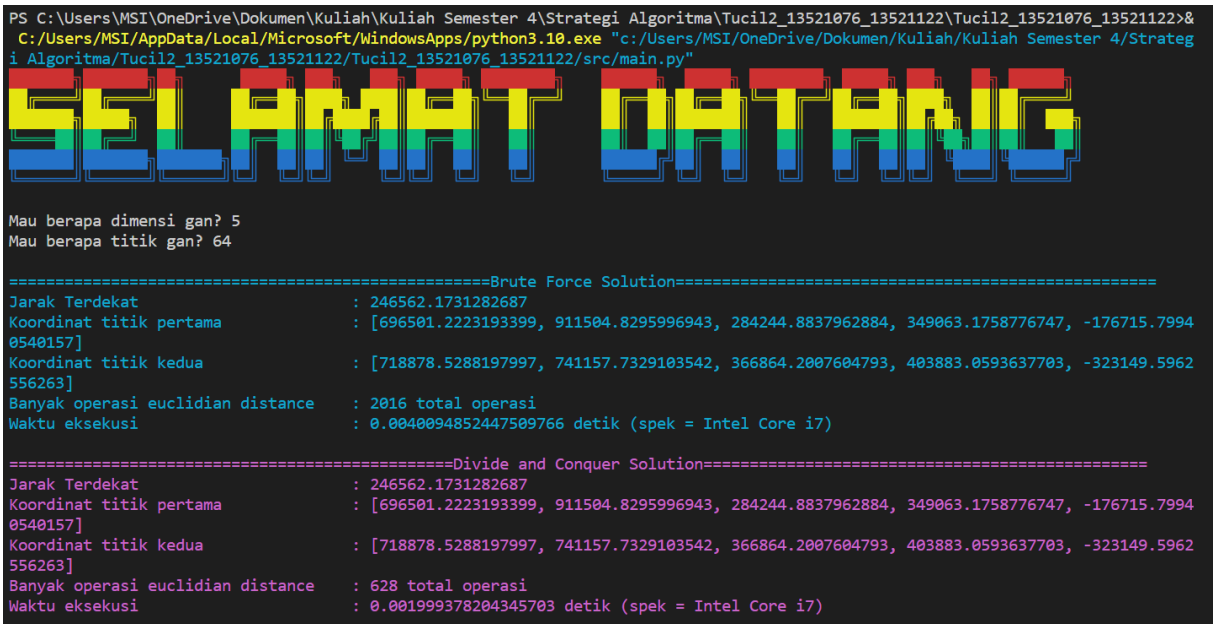


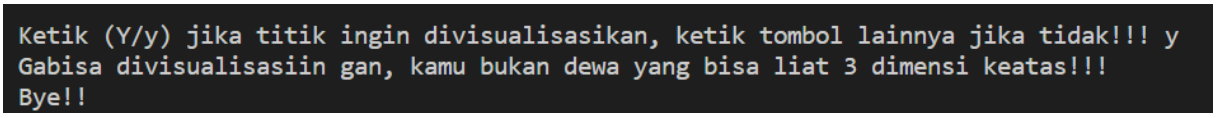

```
Ketik (Y/y) jika titik ingin divisualisasikan, ketik tombol lainnya jika tidak!!! y
Bye!!
```

### 4.2.2 Dimensi 4, n = 128




### 4.2.3 Dimensi 5, n = 64



# BAB V
# KESIMPULAN DAN REFLEKSI

## 5.1 Kesimpulan

Dari program yang kami buat, dapat disimpulkan bahwa algoritma *Divide and Conquer* dapat berjalan lebih cepat daripada algoritma *Brute-force*, serta algoritma *Divide and Conquer* melakukan operasi *euclidean distance* yang lebih sedikit daripada algoritma *Brute-force*. Hal ini dikarenakan algoritma *Divide and Conquer* membagi sebuah *array* (didalam masalah ini adalah daftar titik pada peta) menjadi dua *sub-problems,* lalu dipecah lagi hingga didapatkan hasilnya. Hal ini membuat algoritma *Divide and Conquer* tidak menghitung semua kemungkinan, yang membuatnya memiliki efisiensi O(n). Sedangkan, algoritma *Brute-force* menghitung tiap kemungkinan yang ada, baru setelahnya dicari jarak terpendek, yang membuatnya memiliki efisiensi $O(n^2)$.

## 5.2 Refleksi

Untuk pengerjaan Tugas Kecil 2 IF2211 Strategi Algoritma kali ini, meskipun terdapat beberapa kendala dalam membuat programnya, kami merasa puas dengan hasil program yang telah kami buat.

# BAB VI
# LAMPIRAN

## 6.1 Pranala GitHub

https://github.com/AghnaAbyan/Tucil2_13521076_13521122

## 6.2 *Checklist*

| 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 | ✓ | |