Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Disusun oleh:
Ahmad Ghulam Ilham (13521118)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Tahun 2022/2023

BAB I
Deskripsi Tugas

Mencari sepasang titik terdekat dengan Algoritma Divide and Conquer sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini Anda 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:

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

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
- titik-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)

BAB II
Landasan Teori

Divide and Conquer dulunya adalah strategi militer yang dikenal dengan nama divide ut imperes. Sekarang strategi tersebut menjadi strategi fundamental di dalam ilmu komputer dengan nama Divide and Conquer.

- Divide: membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama)
- Conquer (solve): menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar)
- Combine: mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula

Obyek persoalan yang dibagi : masukan (input) atau instances persoalan yang berukuran n seperti:

- tabel (larik),
- matriks,
- eksponen,
- polinom,
- dll, bergantung persoalannya.

Tiap-tiap upa-persoalan memiliki karakteristik yang sama (the same type) dengan karakteristik persoalan semula sehingga metode Divide and Conquer lebih natural diungkapkan dalam skema rekursif.

Skema Umum Algoritma Divide and Conquer adalah sebagai berikut.

```
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
```

BAB III
Aplikasi Algoritma Divide and Conquer

Algoritma divide and conquer yang digunakan pada implementasi program tucil ini adalah sebagai berikut:

Inisialisasi:

1. Mengurutkan terlebih dahulu himpunan titik berdasarkan komponen absis (x) dari masing-masing titik.

Divide and Conquer:

2. Ambil sebuah bidang melebar sepanjang sumbu y dan sumbu z dengan posisi absis adalah median dari komponen absis semua titik.
3. Bagi himpunan titik P menjadi dua bagian, yaitu P1 (bagian kiri bidang datar) dan P2 (bagian kanan bidang datar).
4. Secara rekursif, kembali lakukan langkah 2 dan 3 sampai mencapai basis.
5. Jika sudah mencapai basis, yaitu ketika jumlah titik dalam himpunan P kurang dari sama dengan 3, gunakan Euclidean Distance untuk mencari jarak terdekat antara titik-titik tersebut.

Combine:

6. Setelah didapat jarak terdekat dari masing-masing himpunan titik (P1 dan P2), akan dibuat sebuah strip yang berisi titik-titik dengan jarak ke bidang datar pemisah himpunan sama dengan jarak terdekat antara P1 atau jarak terdekat antara P2 (digunakan yang lebih kecil).
7. Melakukan perhitungan Euclidean Distance pada titik-titik dalam strip
8. Menggabungkan solusi dengan memilih jarak terdekat antara pasangan titik pada P1, pasangan titik pada P2, atau pasangan titik dalam strip.

Untuk memenuhi algoritma divide and conquer tersebut, berikut adalah tipe data, fungsi, dan prosedur yang digunakan dalam tucil ini.

Tipe data:

Nama	Atribut	Deskripsi
Point	Self.x Self.y Self.z	Tipe data titik pada ruang 3D

Fungsi:

Nama	Parameter Input	Parameter Output	Deskripsi
Bruteforce	Point[]	Float, Point, Point	Mencari pasangan titik terdekat 3D dengan algoritma brute force
Conquer	Point[]	Float, Point, Point	Mencari pasangan titik terdekat 3D dengan

			algoritma divide and conquer
findMidPoint	Point[]	Integer	Mencari posisi median dari himpunan titik
findLeft	Point[], Integer	Point[]	Mencari titik yang berada di himpunan bagian kiri
findRight	Point[], Integer	Point[]	Mencari titik yang berada di himpunan bagian kanan
findMidStrip	Point[], Point[]	Float	Mencari posisi tengah bidang datar strip
Divide	Point[]	Point[], Point[], Float	Membagi himpunan titik P menjadi dua, yaitu P1 (bagian kiri) dan P2 (bagian kanan)
makeStrip	Point[], Float, Float	Point[]	Membuat strip berisi titik yang berada di dekat strip
sortStrip	Point[]	Point[]	Mengurutkan titik dalam strip berdasarkan komponen ordinat (y) dengan menggunakan algoritma quicksort
getInteger	-	Integer	Validasi input nilai n adalah integer
getValid	-	Integer	Validasi input nilai n >= 2
generatePoints	Integer	Point[]	Membangkitkan secara acak titik dalam koordinat (x, y, z)
sortPoints	Point[]	Point[]	Mengurutkan titik berdasarkan komponen absis (x) dengan mneggunakan algoritma quicksort
findDistance	Point, Point	Float	Mencari jarak antara dua buah titik pada bidang 3D dengan Euclidean Distance
findNonStrip	Point[]	Float, Point, Point	Mencari jarak terdekat di antara n buah titik
findInStrip	Point[], Point, Point, Float	Float, Point, Point	Mencari jarak terdekat antara pasangan titik dalam strip
getCount	-	Integer	Menghitung banyak operasi Euclidean Distance

			dilakukan selama program berjalan

Prosedur:

Nama	Parameter Input	Deskripsi
printPoints	Point[]	Menampilkan titik-titik pada bidang 3D ke terminal
displayPoints	Point[], Point[]	Menggambarkan titik dalam bidang 3D

BAB IV
Implementasi dan Pengujian

Implementasi bruteforce.py

bruteforce.py

src > lib > bruteforce.py > ...

You, 1 hour ago | 1 author (You)

```python
from lib.solve import *        You, 1 hour ago • Add program ...

# fungsi mencari pasangan titik terdekat 3D dengan algoritma brute force
def bruteforce(points):
    # memanggil fungsi findNonStrip dari modul solve tanpa memenuhi syarat n <= 3
    return findNonStrip(points)
```

Implementasi conquer.py

conquer.py

src > lib > conquer.py > conquer

You, 1 second ago | 1 author (You)

```python
from lib.divide import *
from lib.solve import *

# fungsi utama mencari pasangan titik terdekat 3D dengan algoritma divide and conquer
def conquer(points):
    # kondisi basis, maka jarak antara titik dihitung langsung dengan rumus Euclidean
    if len(points) <= 3:
        return findNonStrip(points)
    # rekurens, menerapkan algoritma divide and conquer pada masing-masing bagian untuk mencari sepasang titik terdekat
    else:
        leftPoints, rightPoints, middleStrip = divide(points)

        closestLeft, pl1, pl2 = conquer(leftPoints)
        closestRight, pr1, pr2 = conquer(rightPoints)

        if (closestLeft <= closestRight):
            closestDistance = closestLeft
            p1, p2 = pl1, pl2
        else:
            closestDistance = closestRight
            p1, p2 = pr1, pr2

        strip = makeStrip(points, middleStrip, closestDistance)
        strip = sortStrip(strip)

        closestInStrip, ps1, ps2 = findInStrip(strip, p1, p2, closestDistance)

        if (closestDistance <= closestInStrip):
            return closestDistance, p1, p2
        else:
            return closestInStrip, ps1, ps2      You, 1 second ago • Uncommitted changes
```

Implementasi divide.py

divide.py

src > lib > divide.py > ...

```
You, 1 hour ago | 1 author (You)
# fungsi mencari posisi median dari kumpulan titik      You, 1 hour ago • Add program ...
def
    (function) def findLeft(
        points: Any,
        middlePoint: Any
    ) -> Any

# fu                                rada di bagian kiri himpunan
def findLeft(points, middlePoint):
    return points[:middlePoint]

# fungsi mencari titik yang berada di bagian kanan himpunan
def findRight(points, middlePoint):
    return points[middlePoint:]

# fungsi mencari posisi tengah strip
def findMidstrip(leftPoints, rightPoints):
    return (leftPoints[-1].x + rightPoints[0].x) / 2

def divide(points):
        middlePoint = findMidpoint(points)
        leftPoints = findLeft(points, middlePoint)
        rightPoints = findRight(points, middlePoint)
        middleStrip = findMidstrip(leftPoints, rightPoints)
        return leftPoints, rightPoints, middleStrip
```

```
# fungsi membuat strip berisi titik yang berada di dekat strip
def makeStrip(points, middleStrip, closestDistance):
    strip = [p for p in points if abs(p.x - middleStrip) < closestDistance]
    return strip

# fungsi mengurutkan titik dalam strip berdasarkan komponen ordinat (y) dengan menggunakan algoritma quicksort
def sortStrip(strip):
    if len(strip) <= 1:
        return strip
    else:
        pivot = strip[len(strip)//2]      You, 1 hour ago • Add program
        left = [p for p in strip if p.y < pivot.y]
        middle = [p for p in strip if p.y == pivot.y]
        right = [p for p in strip if p.y > pivot.y]
        return sortStrip(left) + middle + sortStrip(right)
```

Implementasi input.py

input.py

src > lib > input.py > ...

You, 31 minutes ago | 1 author (You)

```python
# fungsi validasi input nilai n adalah integer
def getInteger():
    while True:
        try:
            n = int(input("Masukkan nilai n: "))
        except (ValueError):
            print("Input nilai n bukan integer. Ulangi input nilai n!")
            continue
        else:
            break
    return n

# fungsi validasi input nilai n >= 2      You, 31 minutes ago • update program ...
def getValid():
    n = getInteger()
    while (n < 2):
        print("Input nilai (n < 2). Ulangi input nilai n!")
        n = getInteger()
    return n
```

Implementasi point.py

point.py

src > lib > point.py > ...

You, 1 hour ago | 1 author (You)

```python
from numpy import *      You, 1 hour ago • Add program ...
import matplotlib.pyplot as plt

# definisi tipe data point (titik) pada bidang 3D
You, 1 hour ago | 1 author (You)
class Point:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
from matplotlib import *

# fungsi membangkitkan secara acak titik dalam koordinat (x, y, z)
def generatePoints(n):
    points = []
    while len(points) < n:
        x = random.randint(1000)
        y = random.randint(1000)
        z = random.randint(1000)
        p = Point(x, y, z)
        if p not in points:
            points.append(p)
    return points

# fungsi mengurutkan titik berdasarkan komponen absis (x) dengan mneggunakan algoritma quicksort
def sortPoints(points):
    if len(points) <= 1:
        return points
    else:
        pivot = points[len(points)//2] # choose pivot as middle element
        left = [p for p in points if p.x < pivot.x]
        middle = [p for p in points if p.x == pivot.x]
        right = [p for p in points if p.x > pivot.x]
        return sortPoints(left) + middle + sortPoints(right)
```

```python

# prosedur menampilkan titik pada bidang 3D
def printPoints(points):
    for i in range(len(points)):
        print(f"P{i+1} ({points[i].x}, {points[i].y}, {points[i].z})")

# prosedur menggambarkan titik dalam bidang 3D
def displayPoints(points, psol1):
    figure = plt.figure()
    subPlot = figure.add_subplot(projection='3d')

    pxs = [p.x for p in psol1]
    pys = [p.y for p in psol1]
    pzs = [p.z for p in psol1]
    subPlot.scatter(pxs, pys, pzs, color='red')

    pxs = [p.x for p in points]
    pys = [p.y for p in points]
    pzs = [p.z for p in points]
    subPlot.scatter(pxs, pys, pzs, color='black')

    subPlot.set_xlabel('X')
    subPlot.set_ylabel('Y')
    subPlot.set_zlabel('Z')

    plt.show()
```

Implementasi solve.py

solve.py

src > lib > solve.py > getCount

You, 1 hour ago | 1 author (You)

```python
from math import *

countSolve = 0

# fungsi mencari jarak antara dua bauh titik pada bidang 3D
def findDistance(p1, p2):
    global countSolve
    countSolve += 1
    return sqrt((p1.x-p2.x)**2 + (p1.y-p2.y)**2 + (p1.z-p2.z)**2)

# fungsi mencari jarak terdekat di antara n buah titik, dengan n <= 3
def findNonStrip(points):
    closestDistance = inf
    p1, p2 = points[0], points[0]
    for i in range(len(points) - 1):
        for j in range(i+1, len(points)):
            distance = findDistance(points[i], points[j])
            if distance < closestDistance:
                closestDistance = distance
                p1 = points[i]
                p2 = points[j]
    return closestDistance, p1, p2
```

```python

# fungsi mencari jarak terdekat antara pasangan titik dalam strip
def findInStrip(strip, pnon1, pnon2, closesDistance):
    closestInStrip = closesDistance
    p1, p2 = pnon1, pnon2
    for i in range(len(strip)-1):
        for j in range(i+1, len(strip)):
            distance = findDistance(strip[i], strip[j])
            if distance < closestInStrip:
                closestInStrip = distance
                p1 = strip[i]
                p2 = strip[j]
    return closestInStrip, p1, p2

def getCount():
    return countSolve       You, 1 hour ago • Add program …
```

Implementasi main.py

main.py

src > main.py > ...

You, 32 minutes ago | 1 author (You)

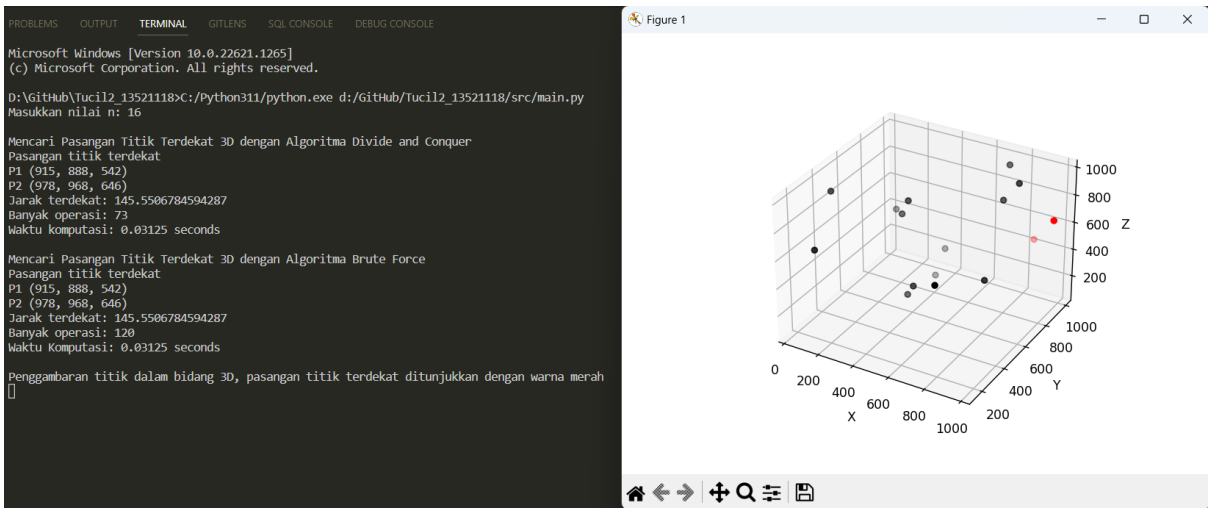
```python
from lib.point import *
from lib.solve import *
from lib.input import *
from lib.conquer import *
from lib.bruteforce import *
from time import *

n = getValid()

points = generatePoints(n)
points = sortPoints(points)

closestDC, pdc1, pdc2 = conquer(points)
timeFinishDC = process_time()
pointsDC = [pdc1, pdc2]
countDC = getCount()

print("\nMencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer")
print("Pasangan titik terdekat")
printPoints(pointsDC)
print(f"Jarak terdekat: {closestDC}")
print(f"Banyak operasi: {countDC}")
print(f"Waktu komputasi: {timeFinishDC} seconds")
```

```python
closestBF, pbf1, pbf2 = bruteforce(points)
timeFinishBF = process_time()
pointsBF = [pbf1, pbf2]
countBF = getCount()-countDC

print("\nMencari Pasangan Titik Terdekat 3D dengan Algoritma Brute Force")
print("Pasangan titik terdekat")
printPoints(pointsBF)
print(f"Jarak terdekat: {closestBF}")
print(f"Banyak operasi: {countBF}")
print(f"Waktu Komputasi: {timeFinishBF} seconds")

print("\nPenggambaran titik dalam bidang 3D, pasangan titik terdekat ditunjukkan dengan warna merah")
points.remove(pdc1)
points.remove(pdc2)
displayPoints(points, pointsDC)      You, 1 hour ago • Add program …
```

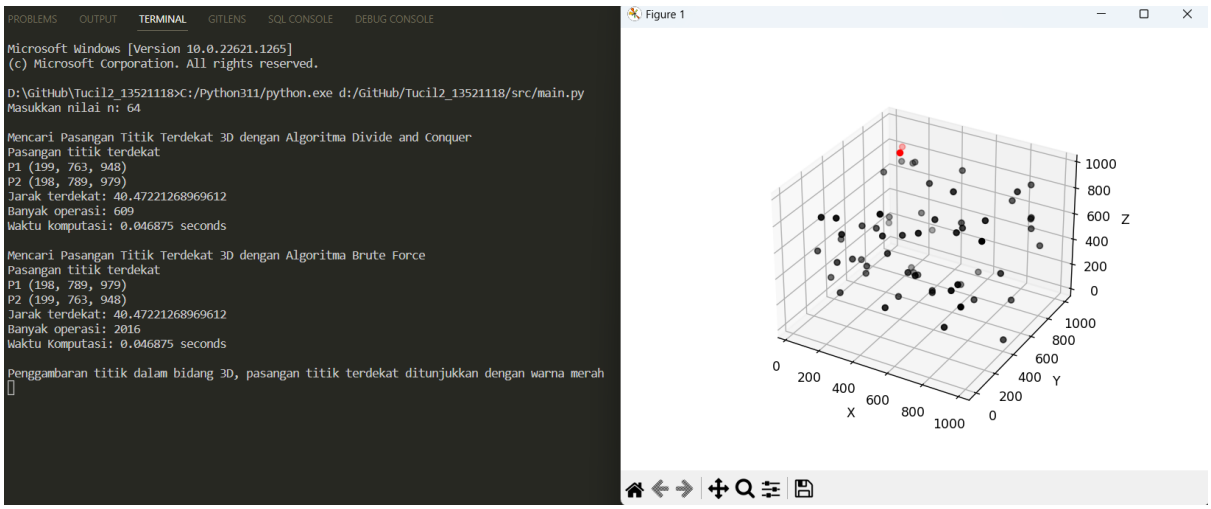
Spesifikasi komputer yang digunakan:

- Processor: AMD Ryzen 5 4600H
- Installed RAM: 16.0 GB

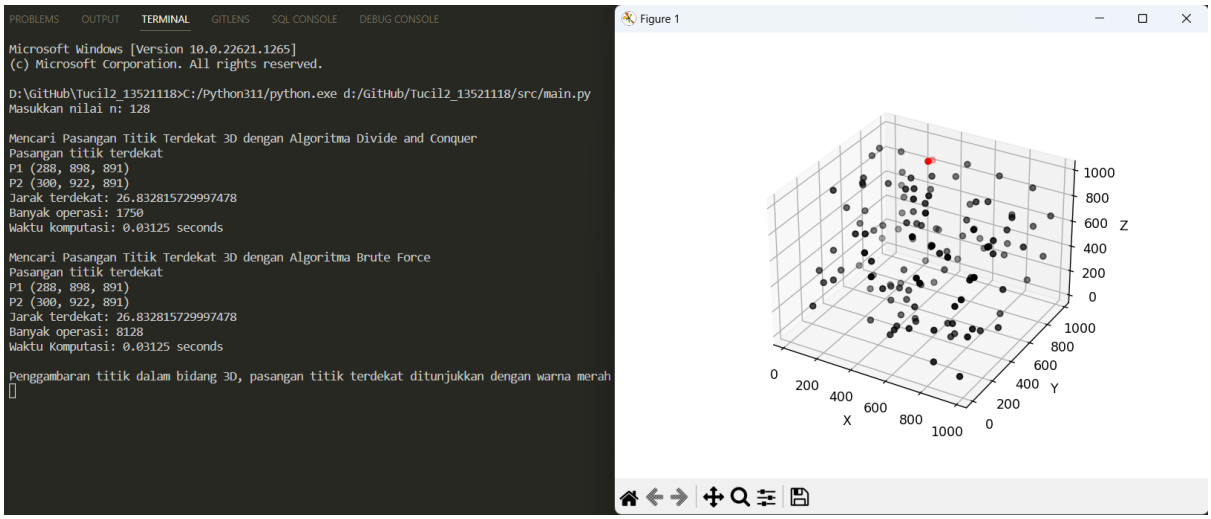
Hasil pengujian dengan n = 16



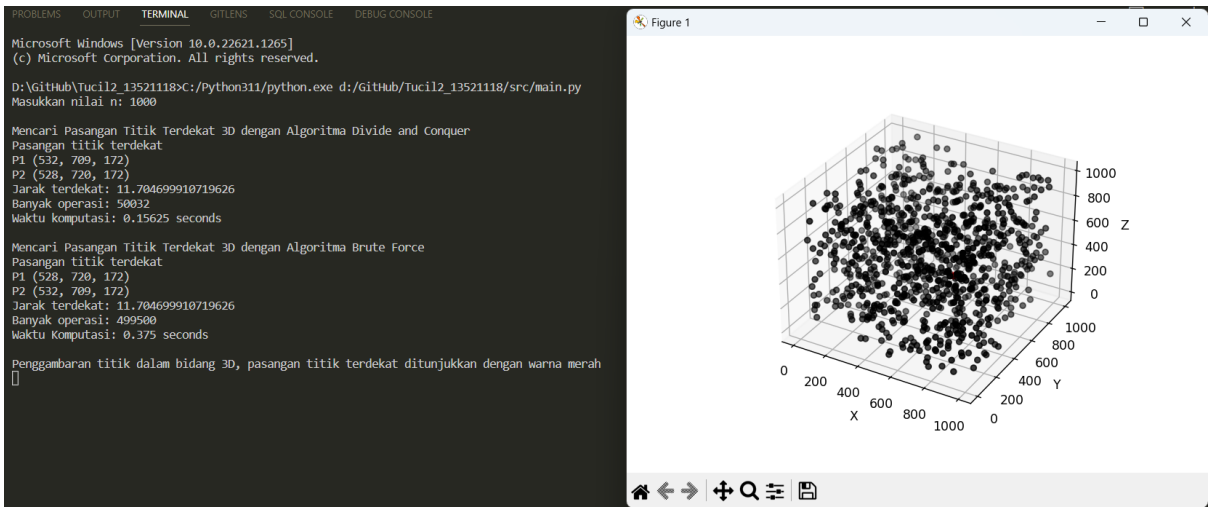
Hasil pengujian dengan n = 64



Hasil pengujian dengan n = 128



Hasil pengujian dengan n = 1000



BAB V
Kesimpulan

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

DAFTAR PUSTAKA

Referensi

- https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf
- https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf

Repository GitHub
https://github.com/Agilham/Tucil2_13521118