

Laporan 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 Tugas 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 titik $P_1 = (x_1, y_1, z_1)$ dan $P_2 = (x_2, y_2, z_2)$ 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 Bahasa C/C++/Java/Python/Golang/Ruby/Perl (pilih salah satu) untuk mencari sepasang titik yang jaraknya terdekat satu 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: menggabungkan 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  $n \leq n_0$  then {ukuran persoalan  $P$  sudah cukup kecil }  
    SOLVE persoalan  $P$  yang berukuran  $n$  ini  
  else  
    DIVIDE menjadi  $r$  upa-persoalan,  $P_1, P_2, \dots, P_r$ , yang masing-masing berukuran  $n_1, n_2, \dots, n_r$   
    for masing-masing  $P_1, P_2, \dots, P_r$ , do  
      DIVIDEandCONQUER( $P_i, n_i$ )  
    endfor  
    COMBINE solusi dari  $P_1, P_2, \dots, P_r$  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 $n \geq 2$
generatePoints	Integer	Point[]	Membangkitkan secara acak titik dalam koordinat (x, y, z)
sortPoints	Point[]	Point[]	Mengurutkan titik berdasarkan komponen absis (x) dengan menggunakan 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 X
src > lib > bruteforce.py > ...
You, 1 hour ago | 1 author (You)
1 from lib.solve import *
2
3 # fungsi mencari pasangan titik terdekat 3D dengan algoritma brute force
4 def bruteforce(points):
5     # memanggil fungsi findNonStrip dari modul solve tanpa memenuhi syarat n <= 3
6     return findNonStrip(points)
```

Implementasi conquer.py

```
conquer.py X
src > lib > conquer.py > conquer
You, 1 second ago | 1 author (You)
1 from lib.divide import *
2 from lib.solve import *
3
4 # fungsi utama mencari pasangan titik terdekat 3D dengan algoritma divide and conquer
5 def conquer(points):
6     # kondisi basis, maka jarak antara titik dihitung langsung dengan rumus Euclidean
7     if len(points) <= 3:
8         return findNonStrip(points)
9     # rekurens, menerapkan algoritma divide and conquer pada masing-masing bagian untuk mencari sepasang titik terdekat
10    else:
11        leftPoints, rightPoints, middleStrip = divide(points)
12
13        closestLeft, p11, p12 = conquer(leftPoints)
14        closestRight, pr1, pr2 = conquer(rightPoints)
15
16        if (closestLeft <= closestRight):
17            closestDistance = closestLeft
18            p1, p2 = p11, p12
19        else:
20            closestDistance = closestRight
21            p1, p2 = pr1, pr2
22
23        strip = makeStrip(points, middleStrip, closestDistance)
24        strip = sortStrip(strip)
25
26        closestInStrip, ps1, ps2 = findInStrip(strip, p1, p2, closestDistance)
27
28        if (closestDistance <= closestInStrip):
29            return closestDistance, p1, p2
30        else:
31            return closestInStrip, ps1, ps2
```

Implementasi divide.py

```
divide.py X
src > lib > divide.py > ...
You, 1 hour ago | 1 author (You)
1 # fungsi mencari posisi median dari kumpulan titik You, 1 hour ago • Add program ...
2 def (function) def findLeft(
3     points: Any,
4     middlePoint: Any
5 ) -> Any
6 # fungsi mencari titik yang berada di bagian kiri himpunan
7 def findLeft(points, middlePoint):
8     return points[:middlePoint]
9
10 # fungsi mencari titik yang berada di bagian kanan himpunan
11 def findRight(points, middlePoint):
12     return points[middlePoint:]
13
14 # fungsi mencari posisi tengah strip
15 def findMidstrip(leftPoints, rightPoints):
16     return (leftPoints[-1].x + rightPoints[0].x) / 2
17
18 def divide(points):
19     middlePoint = findMidpoint(points)
20     leftPoints = findLeft(points, middlePoint)
21     rightPoints = findRight(points, middlePoint)
22     middleStrip = findMidstrip(leftPoints, rightPoints)
23     return leftPoints, rightPoints, middleStrip
24
25 # fungsi membuat strip berisi titik yang berada di dekat strip
26 def makeStrip(points, middleStrip, closestDistance):
27     strip = [p for p in points if abs(p.x - middleStrip) < closestDistance]
28     return strip
29
30 # fungsi mengurutkan titik dalam strip berdasarkan komponen ordinat (y) dengan menggunakan algoritma quicksort
31 def sortStrip(strip):
32     if len(strip) <= 1:
33         return strip
34     else:
35         pivot = strip[len(strip)//2]
36         left = [p for p in strip if p.y < pivot.y]
37         middle = [p for p in strip if p.y == pivot.y]
38         right = [p for p in strip if p.y > pivot.y]
39         return sortStrip(left) + middle + sortStrip(right)
```


Implementasi input.py

```
input.py x
src > lib > input.py > ...
You, 31 minutes ago | 1 author (You)
1 # fungsi validasi input nilai n adalah integer
2 def getInteger():
3     while True:
4         try:
5             n = int(input("Masukkan nilai n: "))
6         except (ValueError):
7             print("Input nilai n bukan integer. Ulangi input nilai n!")
8             continue
9         else:
10            break
11    return n
12
13 # fungsi validasi input nilai n >= 2    You, 31 minutes ago • update program ...
14 def getValid():
15     n = getInteger()
16     while (n < 2):
17         print("Input nilai (n < 2). Ulangi input nilai n!")
18         n = getInteger()
19     return n
20
```

Implementasi point.py

```
point.py x
src > lib > point.py > ...
You, 1 hour ago | 1 author (You)
1 from numpy import *    You, 1 hour ago • Add program ...
2 import matplotlib.pyplot as plt
3
4 # definisi tipe data point (titik) pada bidang 3D
You, 1 hour ago | 1 author (You)
5 class Point:
6     def __init__(self, x, y, z):
7         self.x = x
8         self.y = y
9         self.z = z
10 from matplotlib import *
11
12 # fungsi membangkitkan secara acak titik dalam koordinat (x, y, z)
13 def generatePoints(n):
14     points = []
15     while len(points) < n:
16         x = random.randint(1000)
17         y = random.randint(1000)
18         z = random.randint(1000)
19         p = Point(x, y, z)
20         if p not in points:
21             points.append(p)
22     return points
23
24 # fungsi mengurutkan titik berdasarkan komponen absis (x) dengan menggunakan algoritma quicksort
25 def sortPoints(points):
26     if len(points) <= 1:
27         return points
28     else:
29         pivot = points[len(points)//2] # choose pivot as middle element
30         left = [p for p in points if p.x < pivot.x]
31         middle = [p for p in points if p.x == pivot.x]
32         right = [p for p in points if p.x > pivot.x]
33         return sortPoints(left) + middle + sortPoints(right)
24
```

```

34
35 # prosedur menampilkan titik pada bidang 3D
36 def printPoints(points):
37     for i in range(len(points)):
38         print(f"P{i+1} ({points[i].x}, {points[i].y}, {points[i].z})")
39
40 # prosedur menggambarkan titik dalam bidang 3D
41 def displayPoints(points, psol1):
42     figure = plt.figure()
43     subPlot = figure.add_subplot(projection='3d')
44
45     pxs = [p.x for p in psol1]
46     pys = [p.y for p in psol1]
47     pzs = [p.z for p in psol1]
48     subPlot.scatter(pxs, pys, pzs, color='red')
49
50     pxs = [p.x for p in points]
51     pys = [p.y for p in points]
52     pzs = [p.z for p in points]
53     subPlot.scatter(pxs, pys, pzs, color='black')
54
55     subPlot.set_xlabel('X')
56     subPlot.set_ylabel('Y')
57     subPlot.set_zlabel('Z')
58
59     plt.show()

```

Implementasi solve.py

```

solve.py X
src > lib > solve.py > getCount
You, 1 hour ago | 1 author (You)
1  from math import *
2
3  countSolve = 0
4
5  # fungsi mencari jarak antara dua buah titik pada bidang 3D
6  def findDistance(p1, p2):
7      global countSolve
8      countSolve += 1
9      return sqrt((p1.x-p2.x)**2 + (p1.y-p2.y)**2 + (p1.z-p2.z)**2)
10
11 # fungsi mencari jarak terdekat di antara n buah titik, dengan n <= 3
12 def findNonStrip(points):
13     closestDistance = inf
14     p1, p2 = points[0], points[0]
15     for i in range(len(points) - 1):
16         for j in range(i+1, len(points)):
17             distance = findDistance(points[i], points[j])
18             if distance < closestDistance:
19                 closestDistance = distance
20                 p1 = points[i]
21                 p2 = points[j]
22     return closestDistance, p1, p2
23

```

```

23
24 # fungsi mencari jarak terdekat antara pasangan titik dalam strip
25 def findInStrip(strip, pnon1, pnon2, closesDistance):
26     closestInStrip = closesDistance
27     p1, p2 = pnon1, pnon2
28     for i in range(len(strip)-1):
29         for j in range(i+1, len(strip)):
30             distance = findDistance(strip[i], strip[j])
31             if distance < closestInStrip:
32                 closestInStrip = distance
33                 p1 = strip[i]
34                 p2 = strip[j]
35     return closestInStrip, p1, p2
36
37 def getCount():
38     return countSolve

```

Implementasi main.py

```

main.py X
src > main.py > ...
You, 32 minutes ago | 1 author (You)
1 from lib.point import *
2 from lib.solve import *
3 from lib.input import *
4 from lib.conquer import *
5 from lib.bruteforce import *
6 from time import *
7
8 n = getValid()
9
10 points = generatePoints(n)
11 points = sortPoints(points)
12
13 closestDC, pdc1, pdc2 = conquer(points)
14 timeFinishDC = process_time()
15 pointsDC = [pdc1, pdc2]
16 countDC = getCount()
17
18 print("\nMencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer")
19 print("Pasangan titik terdekat")
20 printPoints(pointsDC)
21 print(f"Jarak terdekat: {closestDC}")
22 print(f"Banyak operasi: {countDC}")
23 print(f"Waktu komputasi: {timeFinishDC} seconds")
24

```

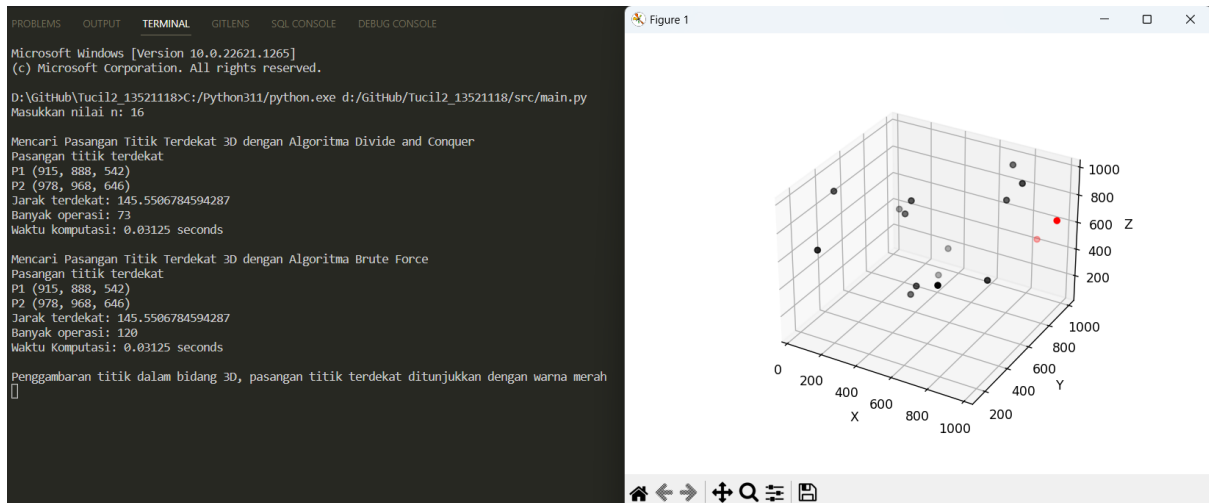
```

24
25 closestBF, pbf1, pbf2 = bruteforce(points)
26 timeFinishBF = process_time()
27 pointsBF = [pbf1, pbf2]
28 countBF = getCount()-countDC
29
30 print("\nMencari Pasangan Titik Terdekat 3D dengan Algoritma Brute Force")
31 print("Pasangan titik terdekat")
32 printPoints(pointsBF)
33 print(f"Jarak terdekat: {closestBF}")
34 print(f"Banyak operasi: {countBF}")
35 print(f"Waktu Komputasi: {timeFinishBF} seconds")
36
37 print("\nPenggambaran titik dalam bidang 3D, pasangan titik terdekat ditunjukkan dengan warna merah")
38 points.remove(pdc1)
39 points.remove(pdc2)
40 displayPoints(points, pointsDC)

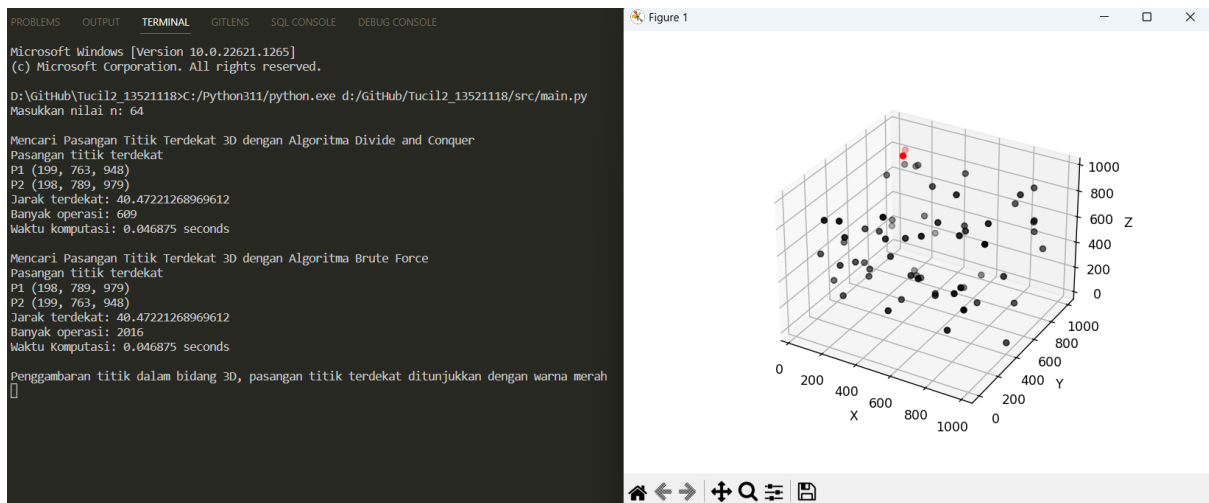
```

You, 1 hour ago • Add program ...

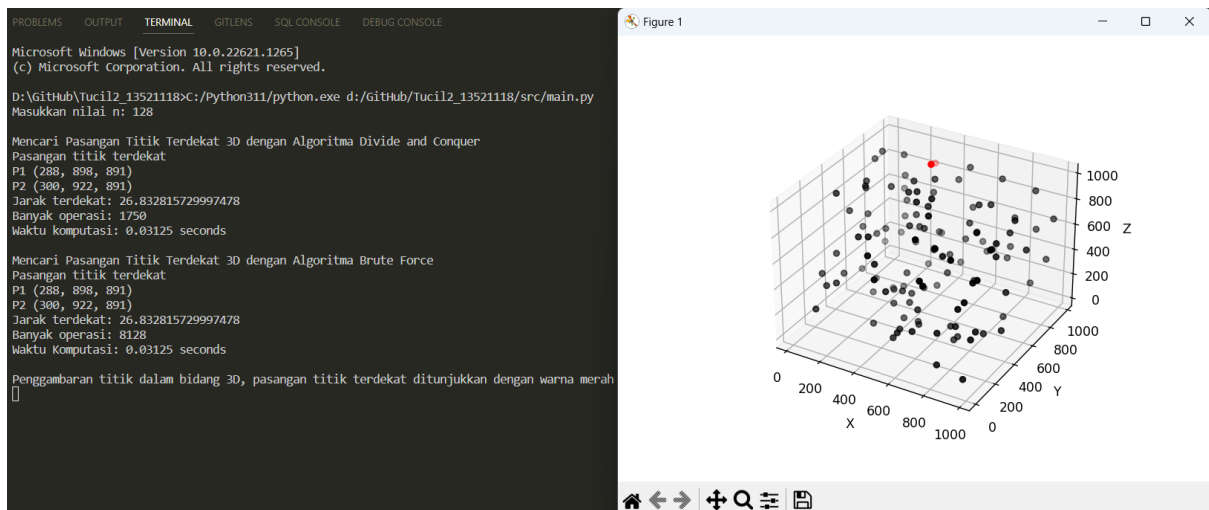
Hasil pengujian dengan n = 16



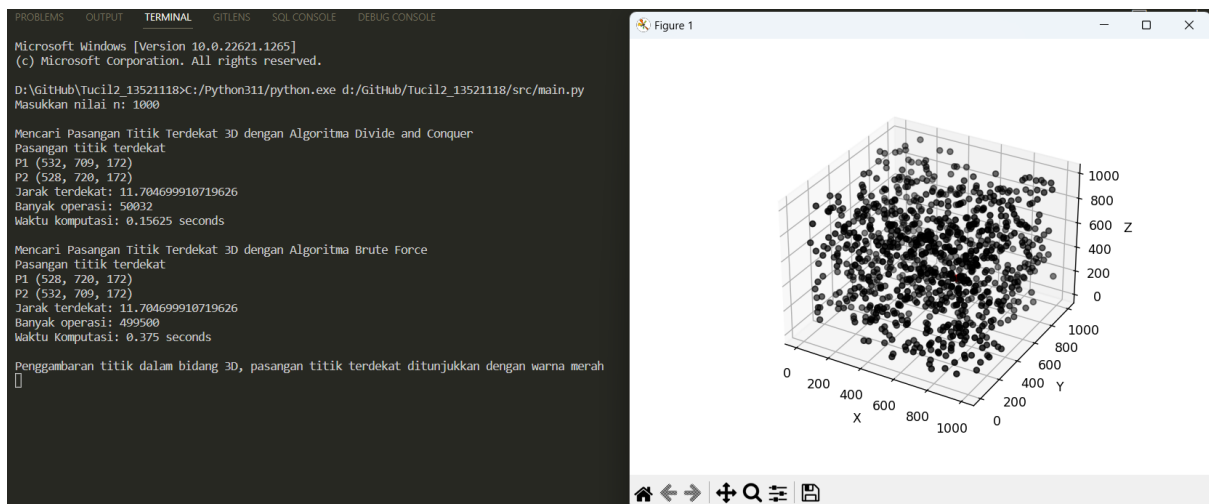
Hasil pengujian dengan n = 64



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 <i>running</i>	✓	
3. Program dapat menerima masukan dan menuliskan luaran	✓	
4. Luaran program sudah benar (solusi <i>closest pair</i> 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)-Bagian1.pdf)Repository
- [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian2.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