Tugas Kecil 2 IF 2211 Strategi Algoritma

Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer*

Hosea Nathanael Abetnego 13521057

Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

I. Deskripsi Persoalan

Mencari sepasang titik terdekat dengan Algoritma *Divide and Conquer* sudah diketahui. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini perlu dilakukan pengembangan 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:

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

Gambar 1 Rumus Euclidean Distance

(dikutip dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil2-Stima-2023.pdf)

II. Algoritma

Program ini menggunakan paradigma algoritma *Divide and Conquer*, yaitu algoritma dengan membagi sebuah persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula dengan ukuran yang lebih kecil sehingga persoalan menjadi lebih kecil dan komputasi lebih sedikit kemudian menggabungkan solusi-solusi tersebut untuk mendapatkan solusi pada permasalahan awal. Langkah-langkah algoritma *Divide and Conquer* adalah:

- *Divide* : membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil
- *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

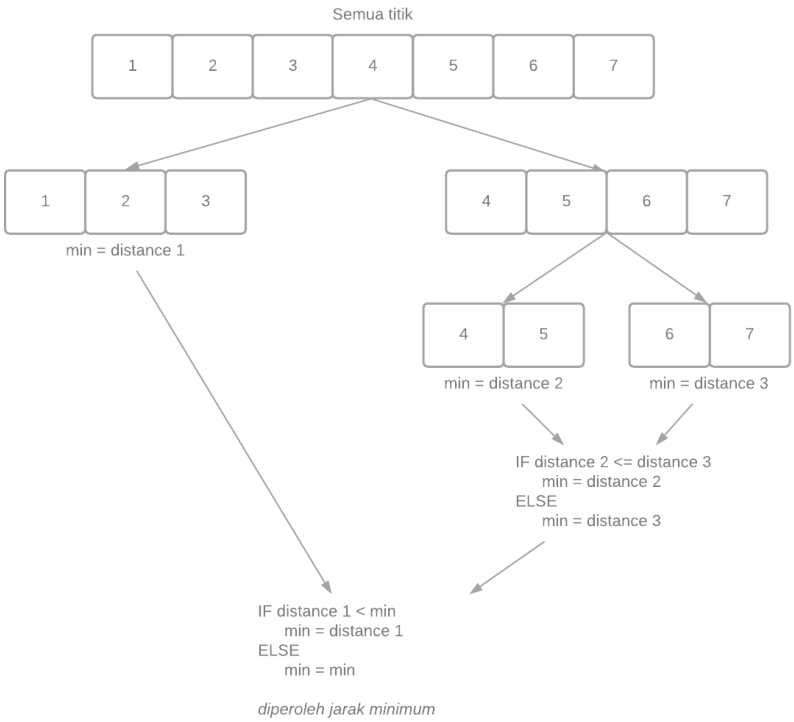
(dikutip dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

Berikut adalah pendekatan yang digunakan pada program ini dalam penyelesaian masalah

1. Program memperoleh n, yaitu banyak titik, dan dimensi dari pengguna
2. Dari banyak titik dan dimensi yang diperoleh, program menentukan secara random posisi titik-titik tersebut dengan batasan -1000 hingga 1000 untuk setiap sumbunya dan disimpan dalam sebuah matriks dengan baris sebagai titik dan kolom sebagai dimensi dan diurutkan dengan algoritma quicksort berdasarkan sumbu x. Contoh:

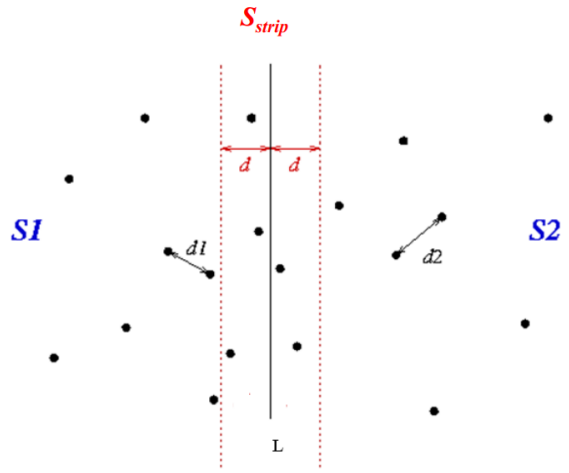
x\y	Sumbu x	Sumbu y	Sumbu z
Titik 1	1	2	3
Titik 2	6	5	4
Titik 3	7	8	9

3. Matriks tersebut kemudian dipecah menjadi 2 upa matriks dengan ukuran yang kurang lebih sama
4. Pemecahan tersebut terus dilakukan hingga pada sebuah matriks hanya terdapat 2 atau 3 titik saja
5. Ketika sudah terdapat 2 titik, jarak terdekat adalah jarak Euclidean antara 2 titik tersebut. Jika terdapat 3 titik, hitung jarak antara semua titik tersebut menggunakan rumus Euclidean dan tentukan yang paling kecil
6. Jarak terkecil yang telah diperoleh tersebut kemudian dibandingkan dengan upa matriks yang lain terus hingga kembali ke matriks utama dan ditentukan jarak terkecilnya. Contoh:



Gambar 2 Penggambaran algoritma Divide and Conquer

7. Karena terdapat kemungkinan terdapat pasangan titik yang jaraknya lebih kecil terdapat diantara pembagian tersebut, lakukan pencarian antara semua titik yang memiliki jarak lebih kecil dari jarak terkecil sebelumnya tersebut pada semua sumbu. Jika ada, maka pasangan titik tersebut menjadi pasangan titik dengan jarak terkecil dan hitung jaraknya



Keterangan: d = MIN($d1$, $d2$)

Gambar 3 Gambaran pencarian jarak titik diantara garis pembagian

8. Lakukan perbandingan antara jarak yang diperoleh pada nomor 6 dan 7 dan tentukan jarak terkecil dan pasangan titiknya
9. Lakukan pencarian jarak terdekat dengan algoritma Brute Force
10. Jarak, titik 1, titik 2, banyak perhitungan Euclidean, dan waktu komputasi dengan *Divide and Conquer* dan *Brute Force* ditampilkan

III. Kode Program

Program terbagi menjadi tiga *file* yaitu main.py, coords.py, dan visualizer.py. main.py berisikan program utama dan fungsi-fungsi untuk menjalankan algoritma *Divide and Conquer* dan juga *Brute Force* pada pencarian jarak. Coords.py berisikan fungsi-fungsi untuk membangkitkan titik-titik, penghitungan Euclidean Distance, algoritma Quicksort, dan pengembalian nilai banyaknya perhitungan Euclidean yang dilakukan. Visualizer.py berisikan fungsi untuk menampilkan hasil perhitungan secara 3D.

Berikut adalah fungsi-fungsi yang terdapat pada main.py:

1. checkAllDimension

   untuk memastikan jarak antara 2 titik lebih kecil dari jarak terkecil pada semua sumbu

```
# Fungsi untuk memastikan jarak antara 2 titik lebih kecil dari jarak terkecil pada semua sumbu
def checkAllDimension(point1, point2, minDis):
    for i in range(len(point1)):
        if (abs(point1[i] - point2[i]) > minDis):
            return False
    return True
```

2. findClosestPairStrip

   untuk mencari jarak terdekat diantara garis pembagi

```
def findClosestPairStrip(points, currMinDis):
    min = currMinDis[0]
```

```
    p1 = currMinDis[1]
    p2 = currMinDis[2]
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            if (checkAllDimension(points[i], points[j], min)):
                distance = coords.EucDis(points[i], points[j])
                if (distance < min):
                    min = distance
                    p1 = points[i]
                    p2 = points[j]

    return (min, p1, p2)
```

3. findClosestPair

   untuk mencari jarak terdekat dengan *Divide and Conquer*

```
def findClosestPair(points):

    if (len(points) == 2):
        return (coords.EucDis(points[0], points[1]), points[0],
points[1])
    elif (len(points) == 3):
        d1 = coords.EucDis(points[0], points[1])
        d2 = coords.EucDis(points[1], points[2])
        d3 = coords.EucDis(points[0], points[2])
        if (d1 >= d2 >= d3):
            return (d3, points[0], points[2])
        elif (d2 >= d1 >= d3):
            return (d3, points[0], points[2])
        elif (d1 >= d3 >= d2):
            return (d2, points[1], points[2])
        elif (d3 >= d1 >= d2):
            return (d2, points[1], points[2])
        elif (d2 >= d3 >= d1):
            return (d1, points[0], points[1])
        elif (d3 >= d2 >= d1):
            return (d1, points[0], points[1])
    else:
        n1 = len(points)//2
        left = points[:n1]
        right = points[n1:]

        d1 = findClosestPair(left)
        d2 = findClosestPair(right)

        if (d1[0] < d2[0]):
            minDistance = d1
        else:
            minDistance = d2

        d3 = findClosestPairStrip(points, minDistance)
        if (d3[0] < minDistance[0]):
            return d3
        else:
            return minDistance
```

4. bruteForceFindClosest

   untuk mencari jarak terdekat dengan algoritma *Brute Force*

```
def bruteForceFindClosest(points):

    min = coords.EucDis(points[0], points[1])
    x = 0
    y = 1
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            distance = coords.bruteEucDis(points[i], points[j])
            if (distance < min):
                min = distance
                x = i
                y = j

    return (min, points[x], points[y])
```

5. main

   untuk memulai program utama

```
def main():
    sys.setrecursionlimit(10000)
    print("")
    print("============================================================
============================")
    print("Mencari Pasangan Titik Terdekat 3D (or more)")
    print("dengan Algoritma Divide and Conquer")
    print("")
    print("By Hosea Nathanael Abetnego - 13521057")
    print("============================================================
============================")
    print("")
    points = coords.getPoints()

    start_time = time.time()
    closestPoints = findClosestPair(coords.quickSort(points))
    end_time = time.time()

    bruteStart_time = time.time()
    bruteClosestPoints = bruteForceFindClosest(points)
    bruteEnd_time = time.time()

    print("")
    print("Results by Divide and Conquer")
    print("Distance                    : ", closestPoints[0])
    print("Point 1                     : ", closestPoints[1])
    print("Point 2                     : ", closestPoints[2])
    print("Banyak perhitungan Euclidean : ", coords.getNumOfEucDis())
```

```
    print("Runtime                      : ", end_time - start_time,
"detik")
    print("")
    print("Results by Brute Force")
    print("Results by Divide and Conquer")
    print("Distance                     : ", bruteClosestPoints[0])
    print("Point 1                      : ", bruteClosestPoints[1])
    print("Point 2                      : ", bruteClosestPoints[2])
    print("Banyak perhitungan Euclidean : ",
coords.getbruteNumEucDis())
    print("Runtime                      : ", bruteEnd_time -
bruteStart_time, "detik")

    if (len(points[0]) == 3):
        print("Visualizing")
        visualizer.visualize(points, closestPoints)
    else:
        print("Dimension not supported, no visualization")
```

6. Algoritma program utama

```
# Memulai program
running = True
while (running):
    main()
    print("")
    again = str(input("Ulangi lagi? (y/n)          : ")).upper()
    if (again == 'N'):
        print("")
        print("Good bye!")
        running = False
```

fungsi-fungsi yang terdapat pada coords.py:

1. ”get” functions

   Memperoleh banyak operasi Euclidean Distance

```python
# Banyaknya penghitungan Euclidean dengan Divide and Conquer dan Brute Force
numOfEucDis = 0    # Deklarasi secara global
bruteNumEucDis = 0 # Deklarasi secara global

# Fungsi untuk mengembalikan banyaknya penghitungan Euclidean dengan Divide and Conquer
def getNumOfEucDis():
    return numOfEucDis

# Fungsi untuk mengembalikan banyaknya penghitungan Euclidean dengan Brute Force
def getbruteNumEucDis():
    return bruteNumEucDis
```

2. ”Euclidean Distance” functions

   Melakukan penghitungan Euclidean Distance untuk algoritma *Divide and Conquer* dan *Brute Force*

```python
# Fungsi mencari Euclidean Distance untuk Divide and Conquer
def EucDis(p1, p2):
    global numOfEucDis

    distance = 0
    for i in range(len(p1)):
        distance += (p2[i] - p1[i])**2

    numOfEucDis += 1
    return distance**(1/2)

# Fungsi mencari Euclidean Distance untuk Brute Force
def bruteEucDis(p1, p2):
    global bruteNumEucDis

    distance = 0
    for i in range(len(p1)):
        distance += (p2[i] - p1[i])**2

    bruteNumEucDis += 1
    return distance**(1/2)
```

3. ”quicksort” functions

   Fungsi untuk memperoleh partisi dan penggabungan pada algoritma quicksort

```python
# Fungsi mempartisi pada algoritma quicksort
def partition(points):
    pivot = points[-1]
    i = 0
    j = len(points) - 1
    while (i < j):
        if (points[i][0] < pivot[0]):
            i += 1
        else:
            j -= 1
            points[i], points[j] = points[j], points[i]
    points[i], points[-1] = points[-1], points[i]
    return points[:i], points[i:]

# Fungsi memulai algoritma quicksort
def quickSort(points):
    if (len(points) <= 1):
        return points
    else:
        leftPart, rightPart = partition(points)
        return quickSort(leftPart) + quickSort(rightPart)
```

4. getPoints

   untuk memperoleh semua titik yang telah dibangkitkan secara random

```python
# Fungsi untuk memperoleh posisi semua titik
def getPoints():

    n = int(input("Masukkan jumlah titik (n)    : "))
    dimension = int(input("Masukkan dimensi titik       : "))

    points = [[0 for i in range(dimension)] for j in range(n)]
    for i in range(n):
        for j in range(dimension):
            points[i][j] = random.uniform(-1000, 1000)

    return quickSort(points)
```

Fungsi pada visualizer.py:

1. visualize

   untuk visualisasi hasil yang diperoleh dan menandakan titik-titik dengan jarak terkecil dengan warna berbeda

```python
# Fungsi visualisasi pada 3D
def visualize(points, closestPair):
    x = points
    p1 = closestPair[1]
    p2 = closestPair[2]

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    for i in range(len(x)):
        ax.scatter(x[i][0], x[i][1], x[i][2], color='b', marker='o')

    ax.scatter(p1[0], p1[1], p1[2], color='r', marker='o')
    ax.scatter(p2[0], p2[1], p2[2], color='r', marker='o')

    ax.set_xlim([-1000, 1000])
    ax.set_ylim([-1000, 1000])
    ax.set_zlim([-1000, 1000])

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

    plt.show()
```

Program ini dapat dijalankan dengan beberapa *requirement*, yaitu:

1. Matplotlib, dan
2. Tkinter

Kemudian, untuk memulai program dapat dilakukan dengan *run* main.py

IV. Percobaan

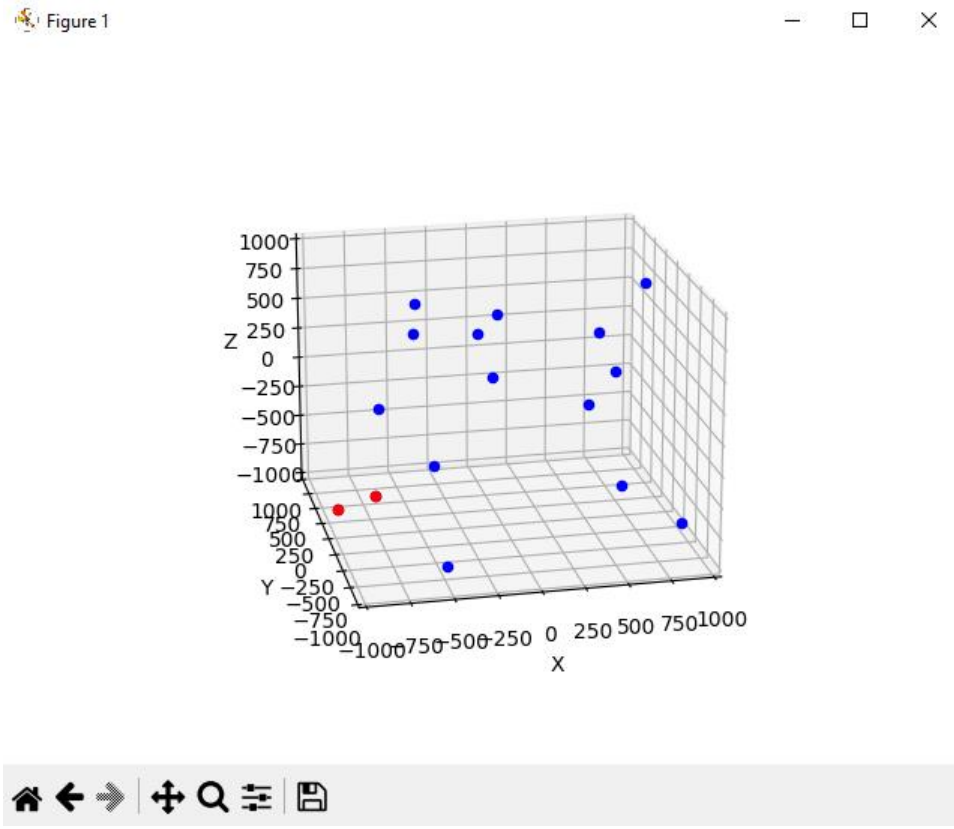
Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada kesalahan	✓	
2. Program berhasil *running*	✓	
3. Program dapat menerima masukan dan menuliskan luaran	✓	
4. Luaran program sudah benar (solusi *closest pair* benar)	✓	
5. Bonus 1 dikerjakan	✓	
6. Bonus 2 dikerjakan	✓	

Berikut adalah beberapa hasil percobaan dari program di atas

1. Percobaan n = 16, dimensi 3

```
================================================================================
Mencari Pasangan Titik Terdekat 3D (or more)
dengan Algoritma Divide and Conquer

By Hosea Nathanael Abetnego - 13521057
================================================================================

Masukkan jumlah titik (n)    : 16
Masukkan dimensi titik       : 3

Results by Divide and Conquer
Distance                     :  326.19893104691596
Point 1                      :  [-986.9270952487459, -26.854440910307062, -791.7542620030636]
Point 2                      :  [-802.0948308315026, -212.21484000651185, -597.1153679108484]
Banyak perhitungan Euclidean :  27
Runtime                      :  0.0 detik

Results by Brute Force
Results by Divide and Conquer
Distance                     :  326.19893104691596
Point 1                      :  [-986.9270952487459, -26.854440910307062, -791.7542620030636]
Point 2                      :  [-802.0948308315026, -212.21484000651185, -597.1153679108484]
Banyak perhitungan Euclidean :  120
Runtime                      :  0.0 detik
Visualizing
```

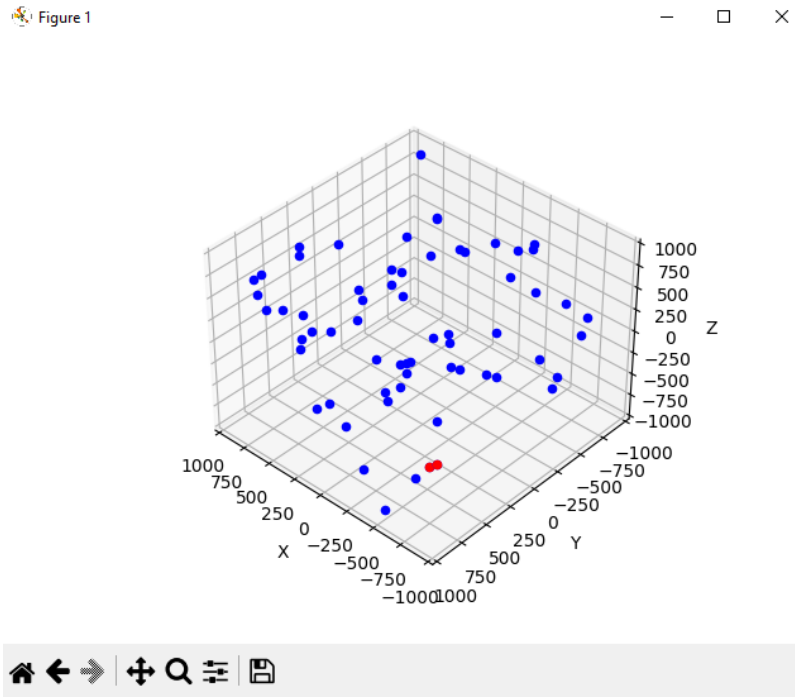
Visualisasi percobaan 1 :



2. Percobaan n = 64, dimensi 3

```
================================================================================
Mencari Pasangan Titik Terdekat 3D (or more)
dengan Algoritma Divide and Conquer

By Hosea Nathanael Abetnego - 13521057
================================================================================

Masukkan jumlah titik (n)    : 64
Masukkan dimensi titik       : 3

Results by Divide and Conquer
Distance                     :  52.977872486958596
Point 1                      :  [-826.2779811589455, 771.0642535271747, -272.98971963608426]
Point 2                      :  [-795.4942669889037, 807.6761446229282, -295.76223257458764]
Banyak perhitungan Euclidean :  110
Runtime                      :  0.001997232437133789 detik

Results by Brute Force
Results by Divide and Conquer
Distance                     :  52.977872486958596
Point 1                      :  [-826.2779811589455, 771.0642535271747, -272.98971963608426]
Point 2                      :  [-795.4942669889037, 807.6761446229282, -295.76223257458764]
Banyak perhitungan Euclidean :  2136
Runtime                      :  0.002001523971557617 detik
Visualizing
```

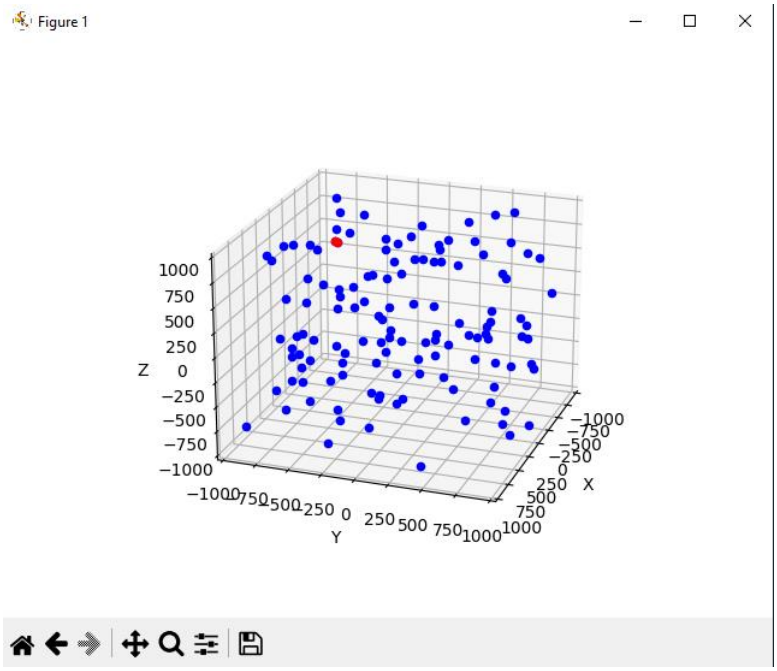
Visualisasi percobaan 2 :



3. Percobaan n = 128, dimensi 3

```
==========================================================================================
Mencari Pasangan Titik Terdekat 3D (or more)
dengan Algoritma Divide and Conquer

By Hosea Nathanael Abetnego - 13521057
==========================================================================================

Masukkan jumlah titik (n)    : 128
Masukkan dimensi titik       : 3

Results by Divide and Conquer
Distance                     :  38.77026565930253
Point 1                      :  [-486.85626167815155, -698.3857200374723, 529.7603618750259]
Point 2                      :  [-467.124387321364, -706.6079871776905, 562.1050825684517]
Banyak perhitungan Euclidean :  281
Runtime                      :  0.0069959163665771484 detik

Results by Brute Force
Results by Divide and Conquer
Distance                     :  38.77026565930253
Point 1                      :  [-486.85626167815155, -698.3857200374723, 529.7603618750259]
Point 2                      :  [-467.124387321364, -706.6079871776905, 562.1050825684517]
Banyak perhitungan Euclidean :  10264
Runtime                      :  0.010000467300415039 detik
Visualizing
```

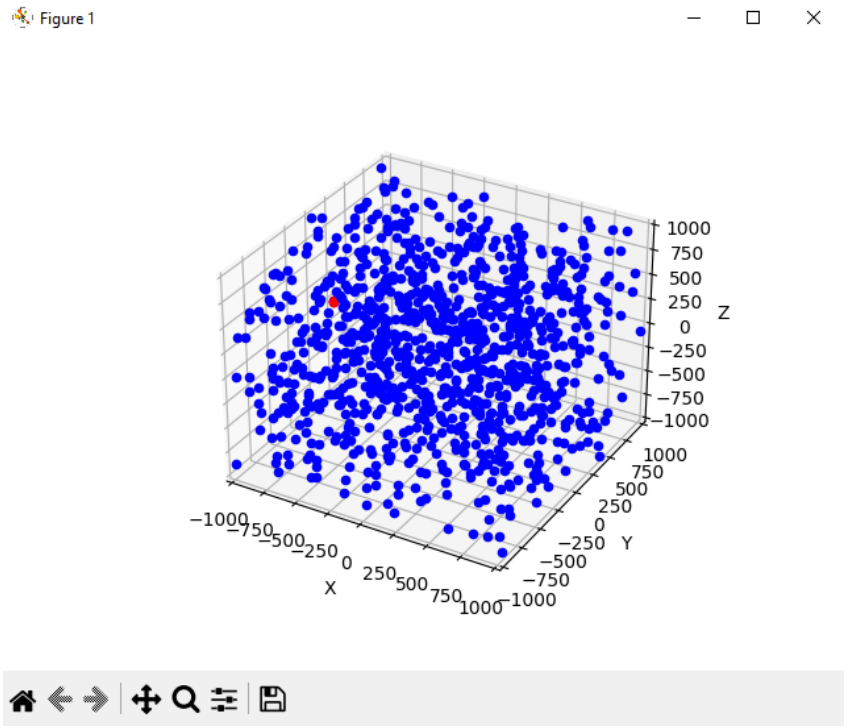
Visualisasi percobaan 3 :



4. Percobaan n = 1000, dimensi 3

```
================================================================================
Mencari Pasangan Titik Terdekat 3D (or more)
dengan Algoritma Divide and Conquer

By Hosea Nathanael Abetnego - 13521057
================================================================================

Masukkan jumlah titik (n)    : 1000
Masukkan dimensi titik       : 3

Results by Divide and Conquer
Distance                     :  13.744848522682759
Point 1                      :  [-623.4571181276433, -320.15560570506545, 490.67197577525053]
Point 2                      :  [-621.7168452666253, -333.1978820373391, 486.6981439527758]
Banyak perhitungan Euclidean :  1729
Runtime                      :  0.40034961700439453 detik

Results by Brute Force
Results by Divide and Conquer
Distance                     :  13.744848522682759
Point 1                      :  [-623.4571181276433, -320.15560570506545, 490.67197577525053]
Point 2                      :  [-621.7168452666253, -333.1978820373391, 486.6981439527758]
Banyak perhitungan Euclidean :  509764
Runtime                      :  0.6219863891601562 detik
Visualizing
```

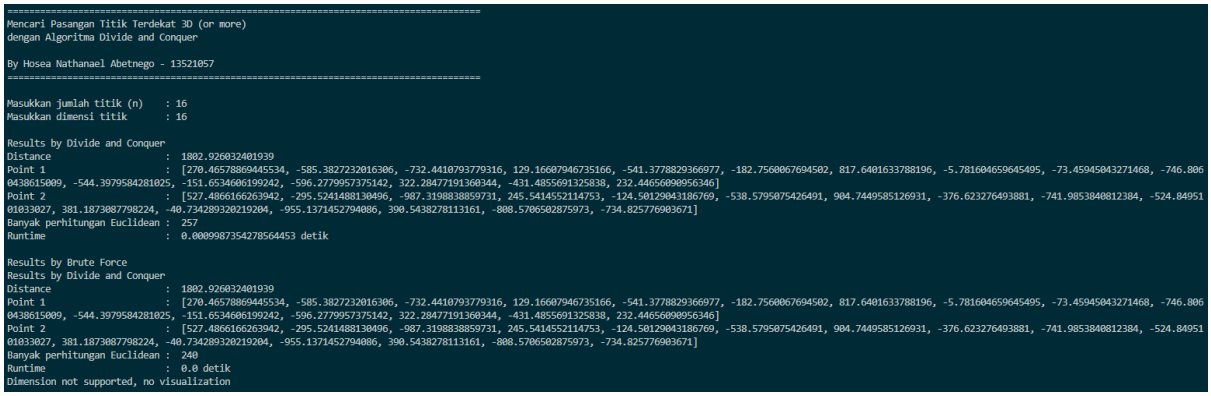
Visualisasi percobaan 4 :



5. Percobaan n = 16, dimensi 5

```
================================================================================
Mencari Pasangan Titik Terdekat 3D (or more)
dengan Algoritma Divide and Conquer

By Hosea Nathanael Abetnego - 13521057
================================================================================

Masukkan jumlah titik (n)    : 16
Masukkan dimensi titik       : 5

Results by Divide and Conquer
Distance                     :  749.5506043993742
Point 1                      :  [230.48444044834105, -159.768846894302, -430.18886333320825, -45.433194453912165, 694.4520121303149]
Point 2                      :  [531.7070458256399, -598.9485008524067, -706.8594996525657, 331.4608026572023, 450.2869940489611]
Banyak perhitungan Euclidean :  50
Runtime                      :  0.00099945068359375 detik

Results by Brute Force
Results by Divide and Conquer
Distance                     :  749.5506043993742
Point 1                      :  [230.48444044834105, -159.768846894302, -430.18886333320825, -45.433194453912165, 694.4520121303149]
Point 2                      :  [531.7070458256399, -598.9485008524067, -706.8594996525657, 331.4608026572023, 450.2869940489611]
Banyak perhitungan Euclidean :  120
Runtime                      :  0.0 detik
Dimension not supported, no visualization
```

6. Percobaan n = 16, dimensi 16



Link Repository

https://github.com/HoseaNA/Tucil2_13521057

Referensi

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf
3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian3.pdf
4. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf
5. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil2-Stima-2023.pdf