Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Dibuat sebagai Salah Satu Luaran Tugas Kecil 2

Matthew Mahendra
13521007

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Daftar Isi

BAB 1

Latar Belakang

1.1 Algoritma *Divide and Conquer*

Algoritma *divide and conquer* berasal dari istilah strategi militer yaitu *divide et impera* yang berarti memecah belah dan menguasai. Algoritma ini menggunakan pendekatan tersebut, yaitu dengan memecah sebuah masalah menjadi masalah-masalah yang kecil (*sub-problems*) lalu menyelesaikan tiap masalah-masalah kecil tersebut. Hasil penyelesaian masalah tersebut kemudian digabungkan untuk mendapatkan hasil akhirnya.

Dari proses penyelesaian algoritma ini, implementasi algoritma *divide and conquer* lebih alami untuk menggunakan penyelesaian secara rekursif. Rekursif dilakukan untuk membagi suatu masalah menjadi permasalahan yang lebih kecil hingga basis yang ditentukan untuk masalah yang sedang diselesaikan.

1.2 Pencarian Titik-Titik Terdekat pada Ruang Dimensi 3

Pada suatu ruang berdimensi 3 yang berisi titik-titik sembarang, menggunakan pendekatan algoritma *divide and conquer*, dapat ditentukan pasangan titik terdekat dari titik-titik tersebut. Perhitungan jarak antara dua titik menggunakan *euclidean distance* (d) untuk ruang n yang dapat dilihat pada rumus 1.1.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 + \dots} \tag{1.1}$$

Selain pada ruang dimensi 3, dapat ditentukan pula pasangan titik tersebut pada ruang berdimensi n dengan $n \geq 2$. Namun, yang dapat digambarkan, untuk saat ini, adalah ruang di dimensi 2 dan 3.

BAB 2

Hasil

2.1 Algoritma yang Digunakan

Algoritma penyelesaian masalah ini adalah sebagai berikut,

1. Untuk titik-titik pada ruang S, urutkan terlebih dahulu menurut absis
2. Setelah diurutkan, pecahlah ruang S menjadi S_1 dan S_2 pada $x_{\frac{n}{2}}$. Isi dari S_1 dan S_2 dengan demikian adalah setengah atau setengah lebih satu dari jumlah titik pada S
3. Bagilah tiap S_1 dan S_2 menjadi ruang yang lebih kecil lagi hingga tersisa 2 atau 3 titik pada masing-masing ruang
4. Selesaikan dengan menghitung euclidean distance dengan kasus sebagai berikut,
   (a) Untuk 2 titik, hitung langsung euclidean distancenya
   (b) Untuk 3 titik, ambil nilai minimal dari kombinasi tiap titik
5. Gabungkan masing-masing S_n, lalu bandingkan hasil jarak dari 2 ruang. Ambillah jarak minimal dari 2 ruang
6. Karena ada kemungkinan bahwa titik terdekat dibatasi oleh partisi, buatlah sebuah daerah s yang memiliki lebar $2d$ dengan pusat pada partisi
7. Dari titik-titik dalam partisi s, tentukan titik-titik yang memiliki jarak lebih kecil daripada jarak minimal yang didapatkan pada langkah 5 dengan melakukan iterasi pada $n, n-1, \ldots, z, y$
8. Lakukan rekursi ke atas hingga masing-masing kembali menjadi ruang S. Jarak terkecil merupakan jarak yang terakhir dikembalikan dari tiap rekursi

2.2 Source Code Program

Program dibuat dengan menggunakan bahasa pemrograman Java versi 19.0.1 dan bahasa pemrograman Python versi 3.11.1. File yang dibuat dengan Java digunakan untuk melakukan proses perhitungan dan rekursi, sedangkan file yang dibuat dengan Python digunakan untuk melakukan visualisasi untuk titik di ruang 2 dan 3.

File-file yang dibuat adalah Main.java, ArrayPoint.java, point.java, ReturnType.java, ShortestDistance.java, dan plot.py. Main.java merupakan program utama (driver). point.java, ReturnType.java, dan ArrayPoint.java berisi *abstract data type* (ADT) yang digunakan pada program ini yaitu ADT titik, ADT senarai titik, dan ADT untuk mengembalikan pasangan titik serta jaraknya. File ShortestDistance.java digunakan untuk menentukan pasangan titik dan jarak terdekatnya. File plot.py digunakan untuk melakukan visualisasi titik-titik di ruang 2 dan 3.

Proses melakukan visualisasi adalah, mengeluarkan derajat, jumlah titik yang dihitung, semua titik di ruang S, dan pasangan titik solusi pada sebuah file teks di folder bin. File ini kemudian akan diproses dengan bahasa pemrograman Python dan divisualisasikan dengan kakas Matplotlib.

Keseluruhan file dapat dijalankan secara bersamaan dengan menggunakan run.bat.

2.2.1 Main.java

```
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args){
        ShortestDistance sd = new ShortestDistance();
        Scanner in = new Scanner(System.in);
        // Make array of points
        System.out.println("Masukkan jumlah titik: ");
        int n = in.nextInt();
        while(n < 2){
            System.out.println("Jumlah titik minimal adalah 2. Silakan ulangi
                masukan!");
            System.out.println("Masukkan jumlah titik: ");
            n = in.nextInt();
        }

        System.out.println("Masukkan derajat titik: ");
        int d = in.nextInt();
        while(d < 2){
            System.out.println("Jumlah derajat terkecil adalah 2. Silakan
               ulangi masukan derajat!");
            System.out.println("Masukkan derajat tiap titik: ");
            d = in.nextInt();
        }

        // Make n points with degree of d
        ArrayPoint ap = new ArrayPoint(n, d);

        // sort point berdasarkan x
        ap.sortArray(ap.array, 0);

        long startDNC = System.currentTimeMillis();
        ReturnType out = sd.findShortestDistance(ap);
        long endDNC = System.currentTimeMillis();

        System.out.println("Shortest Distance is " + out.getDist() + " with
           points " + out.getP1().printPoint() + " and " + out.getP2().
           printPoint());
        System.out.println("Count of Euclidean Distance Operation is " + sd.
           countOfEuc);

        long startB = System.currentTimeMillis();
        System.out.println("Shortest Distance using Brute Force is " + sd.
           findShortestDistanceBrute(ap).getDist() + " with points " + sd.
           findShortestDistanceBrute(ap).getP1().printPoint() + " and " + sd.
           findShortestDistanceBrute(ap).getP2().printPoint());
```

```
        long endB = System.currentTimeMillis();

        /* Print Runtime */
        System.out.println("Divide and Conquer Runtime: " + (endDNC-startDNC)
            + "ms");
        System.out.println("Bruteforce Runtime: " + (endB-startB) + "ms");

        try{
            FileWriter outputFile = new FileWriter("sol.txt");

            outputFile.write(n + "\n");
            outputFile.write(d + "\n");
            if( d== 3 || d == 2){
                for(int i = 0; i < n; i++){
                    if(d == 3){
                        outputFile.write(ap.array[i].buffer[0] + " " + ap.
                           array[i].buffer[1] + " " + ap.array[i].buffer[2] +
                            "\n");
                    }else{
                        outputFile.write(ap.array[i].buffer[0] + " " + ap.
                           array[i].buffer[1] + "\n");
                    }
                }

                if(d == 3){
                    outputFile.write(out.getP1().buffer[0] + " " + out.getP1
                       ().buffer[1] + " " + out.getP1().buffer[2] + "\n");
                }else{
                    outputFile.write(out.getP1().buffer[0] + " " + out.getP1
                       ().buffer[1] + "\n");
                }

                if(d == 3){
                    outputFile.write(out.getP2().buffer[0] + " " + out.getP2
                       ().buffer[1] + " " + out.getP2().buffer[2]);
                }else{
                    outputFile.write(out.getP2().buffer[0] + " " + out.getP2
                       ().buffer[1]);
                }
            }
            outputFile.close();

        } catch (IOException e) {
            System.out.println("Error Occured.");
            e.printStackTrace();
        }
        in.close();
    }
}
```

2.2.2 ArrayPoint.java

```
public class ArrayPoint{
    public int n;
```

```
    public point[] array;

    ArrayPoint(int n, int d){
        this.n = n;
        array = new point[n];
        for(int i = 0; i < n; i++){
            array[i] = new point(d);
        }
    }

    ArrayPoint(int n, point[] array){
        this.n = n;
        this.array = array;
    }

    ArrayPoint(int n){
        this.n = n;
    }

    public void sortArray(point[] arr, int order){
    // Lakukan sorting berdasarkan order (0,1,2,3,...,derajat) (0=x,1=y,2=z
       ,...)
    // Sorting dengan Selection Sort
        for(int i = 0; i < n-1; i++){
            int min = i;
            for(int j = i+1; j < n; j++){
                if(arr[j].buffer[order] < arr[min].buffer[order]){
                    min = j;
                }
            }
            //swap
            point temp_point = arr[min];
            arr[min] = arr[i];
            arr[i] = temp_point;
        }
        this.array = arr;
    }
}
```

2.2.3 point.java

```
import java.util.Random;
class point{
    public int derajat;
    public double[] buffer;
    point(int derajat){
        Random rand = new Random();
        /* Derajat dari tiap titik */
        this.derajat = derajat;

        /* Komponen x,y,z,... dari tiap titik */
        this.buffer = new double[derajat];
        for(int i = 0; i < derajat; i++){
            buffer[i] = rand.nextDouble(100.0);
```

```
        }
    }

    /* Getter untuk x,y,dan z */
    public double getX(){
        return buffer[0];
    }
    public double getY(){
        return buffer[1];
    }
    public double getZ(){
        return buffer[2];
    }

    public String printPoint(){
        String ret = "";
        ret+=("(");
        for(int i = 0; i < derajat; i++){
            if(i != derajat-1)
                ret+=(buffer[i]+",");
            else
                ret+=(buffer[i]);
        }
        ret+=(")");
        return ret;
    }
}
```

2.2.4 ReturnType.java

```
public class ReturnType {
    point p1;
    point p2;
    double dist;

    ReturnType(point p1, point p2, double dist){
        this.p1 = p1;
        this.p2 = p2;
        this.dist = dist;
    }

    public point getP1() {
        return p1;
    }

    public point getP2() {
        return p2;
    }

    public double getDist() {
        return dist;
    }
}
```

2.2.5 ShortestDistance.java

```
import java.lang.Math;
public class ShortestDistance{
    int countOfEuc = 0;

    public double euclideanDistance(point p1, point p2){
        countOfEuc++;
        double buffer = 0;
        for(int i = 0; i < p1.derajat; i++){
            buffer += (p1.buffer[i]-p2.buffer[i]) * (p1.buffer[i]-p2.buffer[i
                ]);
        }
        return Math.sqrt(buffer);
    }

    public ReturnType findShortestDistance(ArrayPoint ap){
        // ap sudah terurut terhadap x
        ReturnType ret;
        if(ap.n == 2){
            ret = new ReturnType(ap.array[0], ap.array[1], euclideanDistance(
                ap.array[0], ap.array[1]));
            return ret;
        }else if(ap.n == 3){
            double t1 = euclideanDistance(ap.array[0], ap.array[1]);
            double t2 = euclideanDistance(ap.array[0], ap.array[2]);
            double t3 = euclideanDistance(ap.array[1], ap.array[2]);
            if(t1 < t2){
                if(t1 < t3){
                    ret = new ReturnType(ap.array[0], ap.array[1], t1);
                }else{
                    ret = new ReturnType(ap.array[1], ap.array[2], t3);
                }
            }else{
                if(t2 < t3){
                    ret = new ReturnType(ap.array[0], ap.array[2], t2);
                }else{
                    ret = new ReturnType(ap.array[1], ap.array[2], t3);
                }
            }
            return ret;
        }else{
            // DIVIDE
            // Define midpoint
            int carry;
            if(ap.n % 2 == 0){
                carry = 0;
            }else{
                carry = 1;
            }

            point[] p1_arr = new point[ap.n/2];
            point[] p2_arr = new point[ap.n/2 + carry];

            for(int i = 0; i < ap.n/2; i++){
                p1_arr[i] = ap.array[i];
```

```
}

for(int i = 0; i < ap.n/2 + carry; i++){
    p2_arr[i] = ap.array[i + ap.n/2];
}

ArrayPoint p1 = new ArrayPoint(ap.n/2, p1_arr);
ArrayPoint p2 = new ArrayPoint(ap.n/2 + carry, p2_arr);

// Combine
ReturnType d1 = findShortestDistance(p1);
ReturnType d2 = findShortestDistance(p2);

// Conquer
if(d1.dist < d2.dist){
    ret = new ReturnType(d1.getP1(), d1.getP2(), d1.dist);
}else{
    ret = new ReturnType(d2.getP1(), d2.getP2(), d2.dist);
}

// Make Slab
int nslab = 0;
for(int i = 0; i < p1.n; i++){
    if(p1.array[i].getX() >= ap.array[ap.n/2].getX()-ret.dist){
        nslab++;
    }
}

for(int i = 0; i < p2.n; i++){
    if(p2.array[i].getX() <= ap.array[ap.n/2].getX()+ret.dist){
        nslab++;
    }
}

ArrayPoint slab = new ArrayPoint(nslab);
slab.array = new point[nslab];
// fill the slab
int nfill = 0;

for(int i = 0; i < p1.n; i++){
    if(p1.array[i].getX() >= ap.array[ap.n/2].getX()-ret.dist){
        slab.array[nfill] = p1.array[i];
        nfill++;
    }
}

for(int i = 0; i < p2.n; i++){
    if(p2.array[i].getX() <= ap.array[ap.n/2].getX()+ret.dist){
        slab.array[nfill] = p2.array[i];
        nfill++;
    }
}

/* Iterasikan untuk jarak terdekat,
 periksa untuk seluruh nilai n, n-1, ..., z, dan y */
```

```
            for(int i = ap.array[0].derajat-1; i > 0 ;i--){
                slab.sortArray(slab.array, i);
                for (int j = 0; j < slab.n; j++){
                    for(int k = j+1; k < slab.n; k++){
                        if(Math.abs(slab.array[j].getX() - slab.array[k].getX
                           ()) < ret.dist || Math.abs(slab.array[j].buffer[i]
                            - slab.array[k].buffer[i]) < ret.dist){
                            ReturnType newret = new ReturnType(slab.array[j],
                                slab.array[k], euclideanDistance(slab.array[j
                               ], slab.array[k]));
                            if(ret.dist > newret.dist){
                                ret = newret;
                            }
                        }
                    }
                }
            }

            return ret;
        }
    }

    public ReturnType findShortestDistanceBrute(ArrayPoint ap){
        // Mencari Shortest Distance Menggunakan Algoritma Bruteforce
        double d = euclideanDistance(ap.array[0], ap.array[1]);
        point p1 = ap.array[0];
        point p2 = ap.array[1];
        for(int i = 0; i < ap.n; i++){
            for(int j = 0; j < ap.n; j++){
                if(i != j){
                    double temp = euclideanDistance(ap.array[i], ap.array[j])
                       ;
                    if(temp < d){
                        d = temp;
                        p1 = ap.array[i];
                        p2 = ap.array[j];
                    }
                }
            }
        }

        ReturnType ret = new ReturnType(p1,p2,d);

        return ret;
    }
}
```

2.2.6 plot.py

```
import argparse
import matplotlib.pyplot as plt
import numpy as np

parser = argparse.ArgumentParser()
```

```
parser.add_argument("path")
args = parser.parse_args()

f = open(args.path, 'r')

# num of points
n = int(f.readline())
# degree of points
d = int(f.readline())

if(d == 2):
# Plot 2D
    # PLOT ALL POINTS
    xpoints = np.array([])
    ypoints = np.array([])
    xpointSol = np.array([])
    ypointSol = np.array([])

    for i in range(n):
        x = f.readline().split()
        for j in range(d):
            x[j] = float(x[j])

        xpoints = np.append(xpoints, x[0])
        ypoints = np.append(ypoints, x[1])

    # GET SOLUTION
    for i in range(2):
        x = f.readline().split()
        for j in range(d):
            x[j] = float(x[j])

        xpointSol = np.append(xpointSol, x[0])
        ypointSol = np.append(ypointSol, x[1])

    plt.plot(xpoints, ypoints, 'ok')
    plt.plot(xpointSol, ypointSol, 'or')
    plt.plot(xpointSol, ypointSol, 'r')
    plt.show()
elif(d==3):
# Plot 3D
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    # PLOT ALL POINTS
    xpoints = np.array([])
    ypoints = np.array([])
    zpoints = np.array([])
    xpointSol = np.array([])
    ypointSol = np.array([])
    zpointsSol = np.array([])

    for i in range(n):
        x = f.readline().split()
        for j in range(d):
```

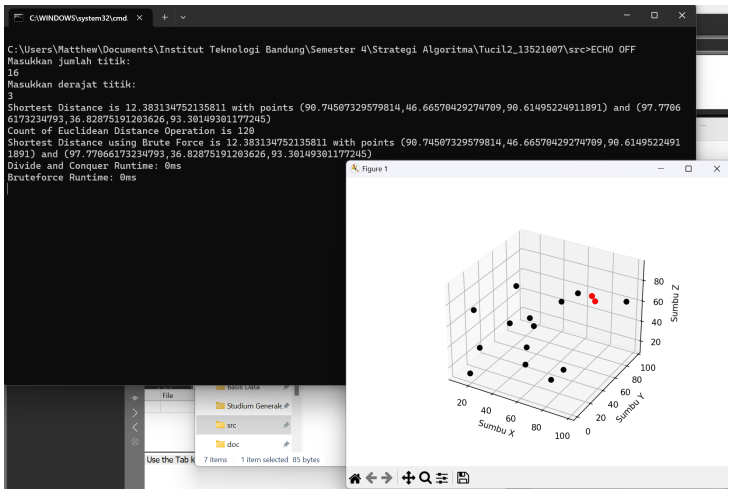
```
            x[j] = float(x[j])

        xpoints = np.append(xpoints, x[0])
        ypoints = np.append(ypoints, x[1])
        zpoints = np.append(zpoints, x[2])

    # GET SOLUTION
    for i in range(2):
        x = f.readline().split()
        for j in range(d):
            x[j] = float(x[j])

        xpointSol = np.append(xpointSol, x[0])
        ypointSol = np.append(ypointSol, x[1])
        zpointsSol = np.append(zpointsSol, x[2])

    for i in range(n):
        ax.plot(xpoints[i], ypoints[i],zpoints[i], 'ok')

    for i in range(2):
        ax.plot(xpointSol[i], ypointSol[i], zpointsSol[i], 'or')

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

    plt.show()

f.close()
```

2.3 Pengujian

Pengujian dilakukan untuk jumlah titik $n = 16, 64, 128, 1000$ dan derajat $d = 2, 3, 4$.

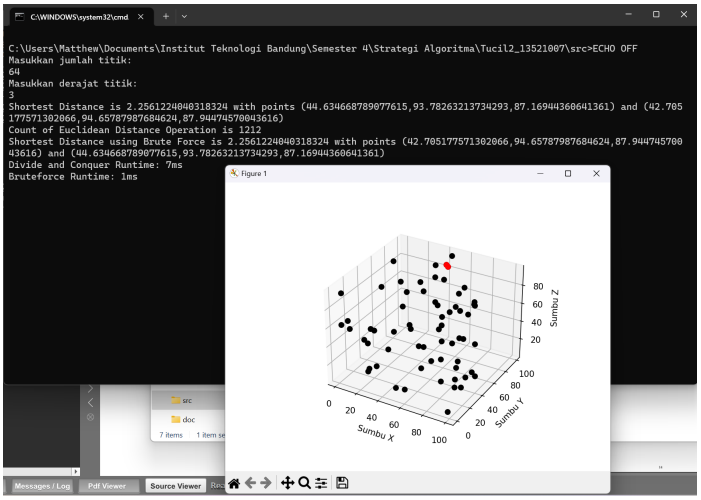
2.3.1 Dimensi 3

16 Titik



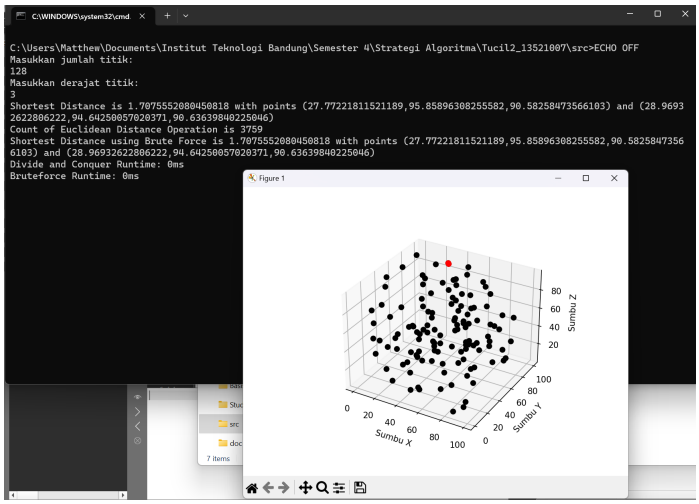
Gambar 2.1: Masukan dan Luaran untuk 16 titik di Ruang Dimensi 3

64 Titik



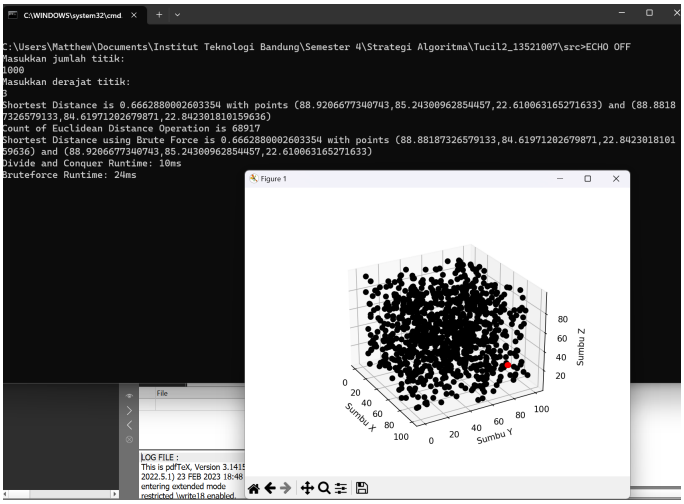
Gambar 2.2: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 3

128 Titik



Gambar 2.3: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 3

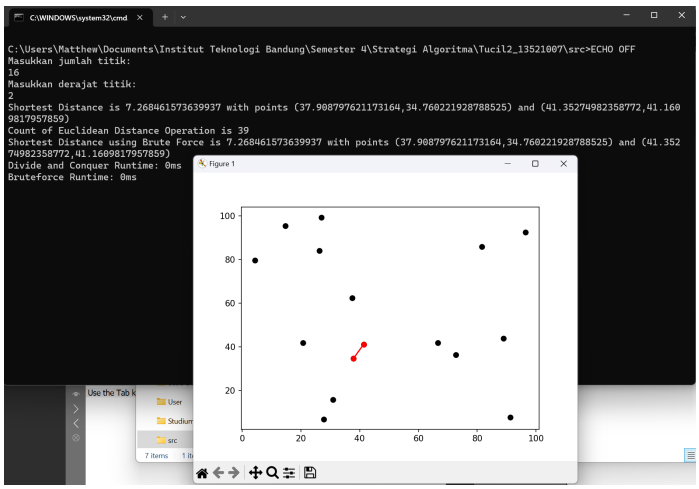
1000 Titik



Gambar 2.4: Masukan dan Luaran untuk 1000 titik di Ruang Dimensi 3

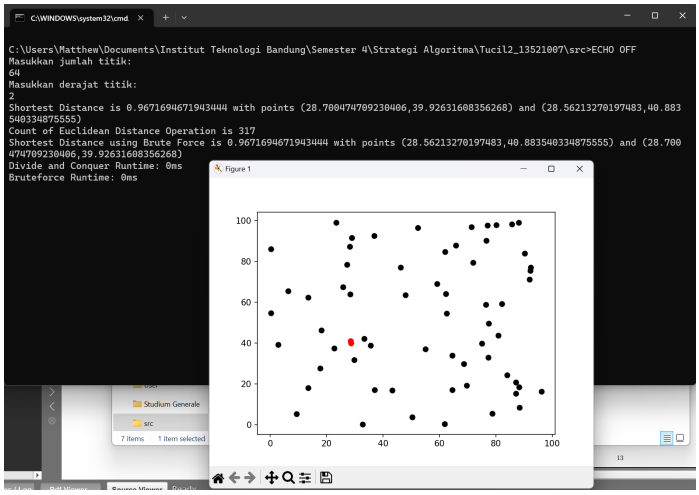
2.3.2 Dimensi 2

16 Titik



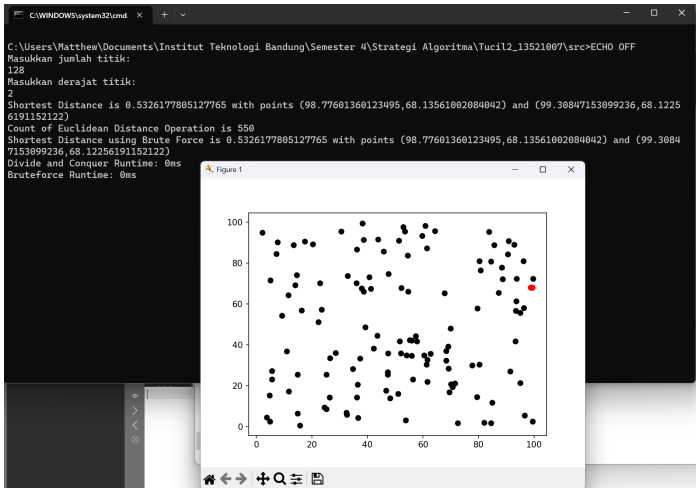
Gambar 2.5: Masukan dan Luaran untuk 16 titik di Ruang Dimensi 2

64 Titik



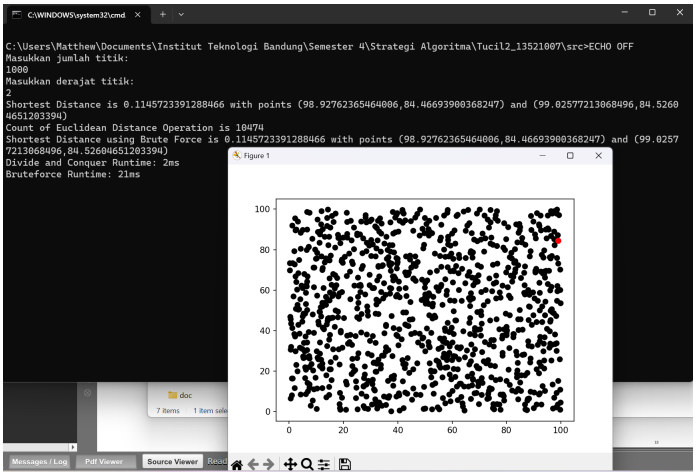
Gambar 2.6: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 2

128 Titik



Gambar 2.7: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 2

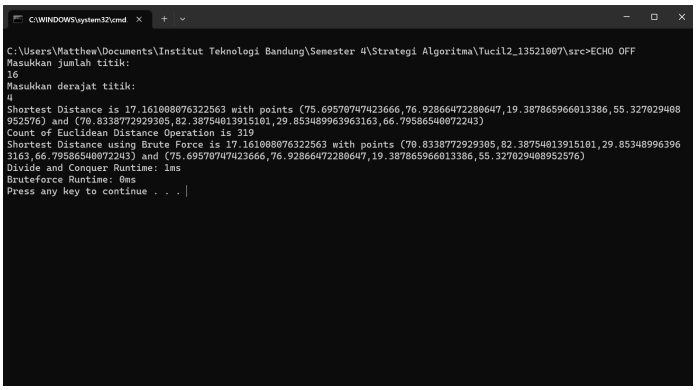
1000 Titik



Gambar 2.8: Masukan dan Luaran untuk 1000 titik di Ruang Dimensi 2

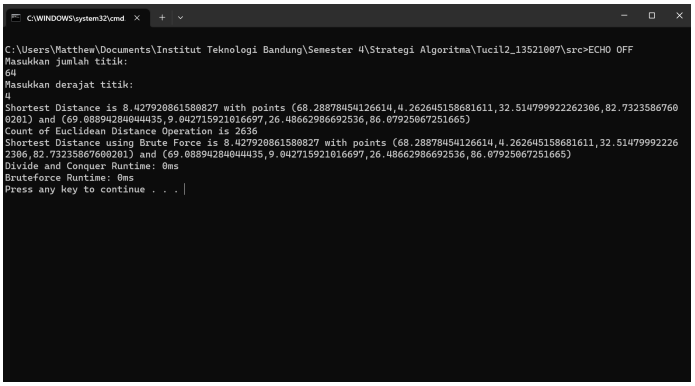
2.3.3 Dimensi 4

16 Titik



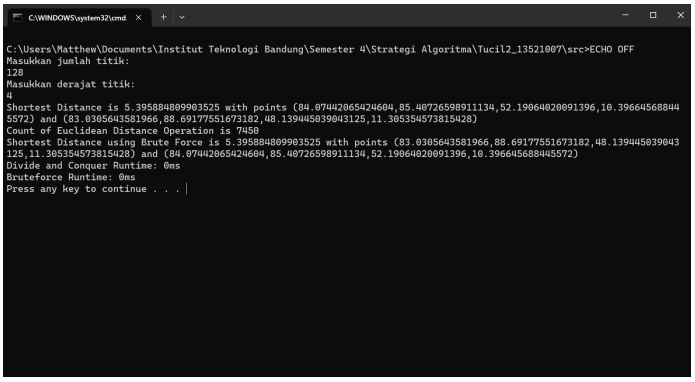
Gambar 2.9: Masukan dan Luaran untuk 16 titik di Ruang Dimensi 4

64 Titik



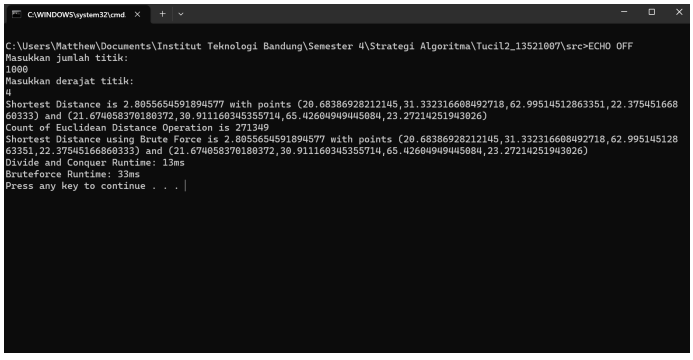
Gambar 2.10: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 4

128 Titik



Gambar 2.11: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 4

1000 Titik



Gambar 2.12: Masukan dan Luaran untuk 1000 titik di Ruang Dimensi 4

BAB 3

Simpulan

3.1 Simpulan

Dari tugas kecil ini, dapat disimpulkan hal-hal sebagai berikut,

1. Algoritma Divide and Conquer dapat diaplikasikan pada permasalahan yang dapat dipecah menjadi persoalan-persoalan yang lebih kecil seperti penentuan titik-titik terdekat

2. Algoritma Divide and Conquer memiliki kompleksitas algoritma yang lebih sederhana dibandingkan algoritma brute force. Hal ini dapat dilihat untuk perbandingan run time pada kasus 1000 titik. Penyelesaian dengan divide and conquer memiliki runtime yang lebih singkat dibandingkan dengan brute force. Oleh sebab itu, untuk mendapatkan hasil pada suatu permasalahan dengan elemen yang banyak dalam waktu yang singkat, algoritma divide and conquer dapat digunakan dibandingkan algoritma brute force

3. Algoritma Divide and Conquer dapat diaplikasikan pada permasalahn pasangan titik terdekat dengan melakukan partisi daerah

Lampiran A

Pranala Github

Tugas ini sudah dipublikasi pada Github dengan pranala `https://github.com/MHEN2606/Tucil2_13521007`

Lampiran B

Check List

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

Tabel B.1: Tabel Check List