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 tersebut dan simpan sebagai δ
6. Karena ada kemungkinan bahwa titik terdekat dibatasi oleh partisi, buatlah sebuah daerah s yang memiliki lebar 2δ 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 Python versi 3.11.1. File yang dibuat adalah Main.py, ShortestDistance.py, dan point.py

Main.py berisi driver dari program. ShortestDistance.py berisi fungsi-fungsi untuk melakukan perhitungan penentuan pasangan titik terdekat. point.py berisi konstruktor titik, konstruktor senarai titik, pengurutan senarai, dan mengeluarkan string titik.

Visualisasi dibuat dengan mempertimbangkan dimensi dari titik, keseluruhan titik yang digunakan, dan pasangan titik terdekat. Visualisasi untuk ruang dimensi 2 dan 3 menggunakan bantuan kakas matplotlib.

2.2.1 Main.py

```
# Filename: Main.py
# Matthew Mahendra - 13521007
import point as pt
import ShortestDistance as sd
import time as tm
import numpy as np
import matplotlib.pyplot as plt

### HEADER ###
print("========================")
print(" CLOSEST PAIR OF POINTS ")
print("========================")

# Input Program
n = int(input("Masukkan jumlah titik: "))

while(n < 2):
    print("Jumlah titik minimal adalah 2. Silakan ulangi masukan!")
    print("Masukkan jumlah titik: ")
    n = int(input("Masukkan jumlah titik: "))

d = int(input("Masukkan dimensi titik: "))

while(d < 2):
    print("Dimensi terkecil adalah 2. Silakan ulangi masukan derajat!")
    d = int(input("Masukkan dimensi titik: "))

# Make Points
ap = pt.makeArrayPoint(n,d)

# sort point berdasarkan x
ap = pt.sortArray(ap, 0)

startDNC = tm.time()
out = sd.findShortestDistance(ap, d)
endDNC = tm.time()
coe = sd.countOfEuc
print("========================")
print("        RESULT          ")
print("========================")
print("USING DIVIDE AND CONQUER")
print("========================")
print("Shortest Distance is", round(out[2],2), "with points", pt.printPoint(
   out[0], d), "and", pt.printPoint(out[1], d))
print("Euclidean Distance operations count:", coe)

startB = tm.time()
brute = sd.findShortestDistanceBrute(ap)
endB = tm.time()
print("========================")
print("   USING BRUTE FORCE")
print("========================")
print("Shortest Distance using Brute Force is", round(brute[2],2), "with
   points", pt.printPoint(brute[0], d), "and", pt.printPoint(brute[1], d))
```

```
print("Euclidean Distance operations count:", (sd.countOfEuc-coe))

# Print Runtime
print("========================")
print("        RUNTIME         ")
print("========================")
print("Divide and Conquer Runtime:", round((endDNC-startDNC)*1000,2), "ms")
print("Bruteforce Runtime:", round((endB-startB)*1000,2), "ms")

# Visualization
if(d == 2 or d == 3):
    choice = input("Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y
       /n) ")
    if(choice == "y"):
        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):
                xpoints = np.append(xpoints, ap[i][0])
                ypoints = np.append(ypoints, ap[i][1])

            # GET SOLUTION
            for i in range(2):
                xpointSol = np.append(xpointSol, out[i][0])
                ypointSol = np.append(ypointSol, out[i][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):
                xpoints = np.append(xpoints, ap[i][0])
                ypoints = np.append(ypoints, ap[i][1])
                zpoints = np.append(zpoints, ap[i][2])

            # GET SOLUTION
            for i in range(2):
```

```
                xpointSol = np.append(xpointSol, out[i][0])
                ypointSol = np.append(ypointSol, out[i][1])
                zpointsSol = np.append(zpointsSol, out[i][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()
```

2.2.2 point.py

```
# Filename: point.py
# Matthew Mahendra - 13521007
import random

# Konstruktor point
# Point adalah array of float
def makePoint(d):
    point = [random.uniform(0,101) for i in range(d)]
    return point

# Konstruktor set of point
# Set adalah array of array of float
def makeArrayPoint(n,d):
    arrayPoint = [makePoint(d) for i in range(n)]
    return arrayPoint

# Fungsi sorting array menggunakan selection sort
def sortArray(arr, order):
    # Lakukan sorting berdasarkan order (0,1,2,3,...,derajat) (0=x,1=y,2=z
       ,...)
    # Sorting dengan Selection Sort
    for i in range(len(arr)-1):
        min = i
        for j in range(i+1, len(arr)):
            if(arr[j][order] < arr[min][order]):
                min = j

        temp_point = arr[min]
        arr[min] = arr[i]
        arr[i] = temp_point

    return arr

# Fungsi untuk return string terformat
def printPoint(buffer,d):
    ret = ""
```

```
    ret+=("(")
    for i in range(d):
        if(i != d-1):
            ret+=str(round(buffer[i],2))+(",")
        else:
            ret+=str(round(buffer[i],2))
    ret+=(")")
    return ret
```

2.2.3 ShortestDistance.py

```
# Filename: ShortestDistance.py
# Matthew Mahendra - 13521007
import point as pt
import math

countOfEuc = 0

# Fungsi untuk menentukan euclidean distance antar dua titik
def euclideanDistance(p1, p2):
    global countOfEuc
    countOfEuc += 1
    d = len(p1)
    buffer = 0.0
    for i in range(d):
        buffer+=(p1[i]-p2[i])*(p1[i]-p2[i])
    return math.sqrt(buffer)

# Fungsi untuk menentukan jarak terdekat dari set of points
def findShortestDistance(ap, d):
    # Jumlah titik dengan menghitung panjang dari array of points
    n = len(ap)
    # Basis ketika jumlah titik adalah 2 atau 3
    if(n == 2):
        return [ap[0],ap[1], euclideanDistance(ap[0], ap[1])]
    elif (n==3):
        t1 = euclideanDistance(ap[0], ap[1])
        t2 = euclideanDistance(ap[0], ap[2])
        t3 = euclideanDistance(ap[1], ap[2])

        if(t1 < t2):
            if(t1 < t3):
                ret = [ap[0], ap[1], t1]
            else:
                ret = [ap[1], ap[2], t3]

        else:
            if(t2 < t3):
                ret = [ap[0], ap[2], t2]
            else:
                ret = [ap[1], ap[2], t3]

        return ret
    else:
```

```
    # Divide
    if(n % 2 == 0):
        carry = 0
    else:
        carry = 1

    # Bagi titik-titik menjadi partisi p1 dan p2
    p1 = [[] for i in range (n//2)]
    p2 = [[] for i in range ((n//2)+carry)]
    for i in range(n//2):
        p1[i] = ap[i]

    for i in range((n//2) + carry):
        p2[i] = ap[i + (n//2)]

    # Cari jarak minimum pada tiap partisi
    d1 = findShortestDistance(p1, d)
    d2 = findShortestDistance(p2, d)

    # Conquer
    # Bandingkan d1 dan d2 untuk mengambil minimumnya
    if(d1[2] < d2[2]):
        ret = [d1[0], d1[1], d1[2]]
    else:
        ret = [d2[0], d2[1], d2[2]]

    # Combine
    # Make a Slab

    # Hitung jumlah yang akan dimasukkan ke dalam slab
    nslab = 0
    for i in range(len(p1)):
        if(p1[i][0] >= ap[n//2][0]-ret[2]):
            nslab += 1

    for i in range(len(p2)):
        if(p2[i][0] <= ap[n//2][0]+ret[2]):
            nslab += 1

    slab = [[] for i in range(nslab)]

    nfill = 0
    for i in range(len(p1)):
        if(p1[i][0] >= ap[n//2][0]-ret[2]):
            slab[nfill] = p1[i]
            nfill += 1

    for i in range(len(p2)):
        if(p2[i][0] <= ap[n//2][0]+ret[2]):
            slab[nfill] = p2[i]
            nfill += 1

    for i in range(d-1, 0,-1):
        slab = pt.sortArray(slab, i)
        for j in range(len(slab)):
```

```
                for k in range(j+1, len(slab)):
                    if(abs(slab[j][0] - slab[k][0]) < ret[2] or abs(slab[j][i
                        ] - slab[k][i]) < ret[2]):
                        newret = [slab[j], slab[k], euclideanDistance(slab[j
                            ], slab[k])]
                        if(ret[2] > newret[2]):
                            ret = newret

        return ret

def findShortestDistanceBrute(ap):
    # Mencari Shortest Distance Menggunakan Algoritma Bruteforce
    d = euclideanDistance(ap[0], ap[1])
    p1 = ap[0]
    p2 = ap[1]
    for i in range(len(ap)):
        for j in range(i+1, len(ap)):
            temp = euclideanDistance(ap[i], ap[j])
            if(temp < d):
                d = temp
                p1 = ap[i]
                p2 = ap[j]

    ret = [p1,p2,d]

    return ret
```

2.3 Pengujian

Pengujian dilakukan untuk jumlah titik $n = 16, 64, 128, 1000$ dan dimensi $d = 2, 3, 4$. Pengujian dilakukan pada perangkat dengan sistem operasi Windows 11, prosesor Intel i7-11370H, dan memori 16 GB.

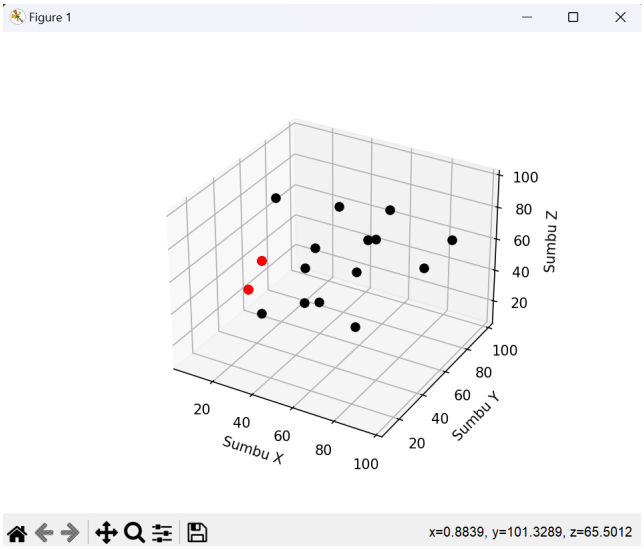
Pengujian akan menampilkan jarak pasangan titik terdekat dan pasangan titik terdekat dengan algoritma *divide and conquer* sekaligus jarak pasangan titik terdekat dan pasangan titik terdekat dengan algoritma *brute force*. Pada akhir program akan ditampilkan perbandingan waktu kompilasi dari masing-masing algoritma yang digunakan pada program.

2.3.1 Dimensi 3

16 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 16
Masukkan dimensi titik: 3
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 15.76 with points (12.99,55.84,45.13) and (13.88,43.51,35.36)
Euclidean Distance operations count: 138
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 15.76 with points (12.99,55.84,45.13) and (13.88,43.51,35.36)
Euclidean Distance operations count: 121
========================
        RUNTIME
========================
Divide and Conquer Runtime: 0.0 ms
Bruteforce Runtime: 0.0 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

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

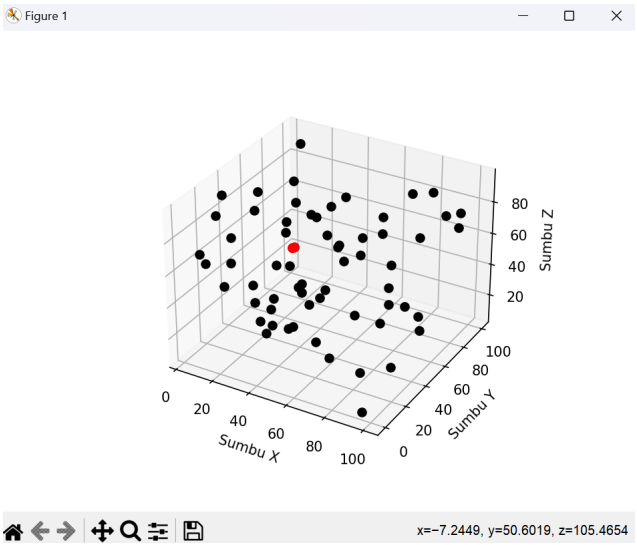


Gambar 2.2: Visualisasi Hasil untuk 16 titik di Ruang Dimensi 3

64 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
=======================
 CLOSEST PAIR OF POINTS
=======================
Masukkan jumlah titik: 64
Masukkan dimensi titik: 3
=======================
        RESULT
=======================
USING DIVIDE AND CONQUER
=======================
Shortest Distance is 1.44 with points (49.84,16.92,82.14) and (51.08,16.99,82.88)
Euclidean Distance operations count: 830
=======================
   USING BRUTE FORCE
=======================
Shortest Distance using Brute Force is 1.44 with points (49.84,16.92,82.14) and (51.08,16.99,82.88)
Euclidean Distance operations count: 2017
=======================
        RUNTIME
=======================
Divide and Conquer Runtime: 1.09 ms
Bruteforce Runtime: 2.46 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.3: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 3

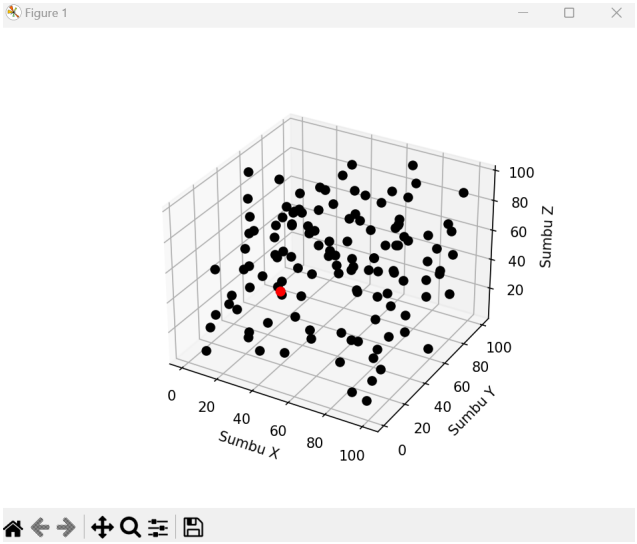


Gambar 2.4: Visualisasi Hasil untuk 64 titik di Ruang Dimensi 3

128 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 128
Masukkan dimensi titik: 3
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 0.5 with points (5.08,75.53,0.57) and (5.14,75.3,1.02)
Euclidean Distance operations count: 2848
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 0.5 with points (5.08,75.53,0.57) and (5.14,75.3,1.02)
Euclidean Distance operations count: 8129
========================
        RUNTIME
========================
Divide and Conquer Runtime: 3.69 ms
Bruteforce Runtime: 3.03 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.5: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 3

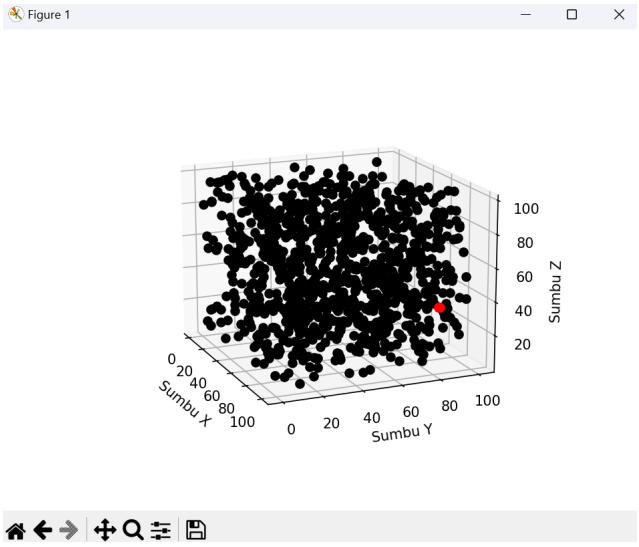


Gambar 2.6: Visualisasi Hasil untuk 128 titik di Ruang Dimensi 3

1000 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
=======================
 CLOSEST PAIR OF POINTS
=======================
Masukkan jumlah titik: 1000
Masukkan dimensi titik: 3
=======================
        RESULT
=======================
USING DIVIDE AND CONQUER
=======================
Shortest Distance is 0.9 with points (95.26,85.33,35.87) and (95.41,84.57,36.32)
Euclidean Distance operations count: 86130
=======================
   USING BRUTE FORCE
=======================
Shortest Distance using Brute Force is 0.9 with points (95.26,85.33,35.87) and (95.41,84.57,36.32)
Euclidean Distance operations count: 499501
=======================
        RUNTIME
=======================
Divide and Conquer Runtime: 55.73 ms
Bruteforce Runtime: 226.58 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.7: Masukan dan Luaran untuk 1000 titik di Ruang Dimensi 3



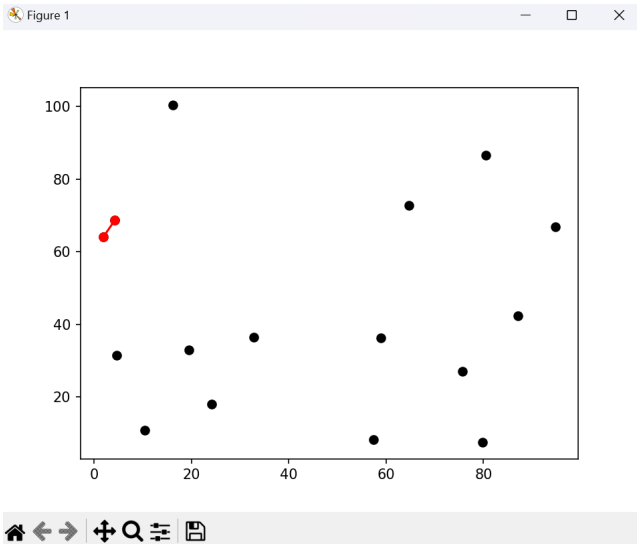
Gambar 2.8: Visualisasi Hasil untuk 1000 titik di Ruang Dimensi 3

2.3.2 Dimensi 2

16 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
=======================
 CLOSEST PAIR OF POINTS
=======================
Masukkan jumlah titik: 16
Masukkan dimensi titik: 2
=======================
        RESULT
=======================
USING DIVIDE AND CONQUER
=======================
Shortest Distance is 5.25 with points (1.86,64.04) and (4.26,68.71)
Euclidean Distance operations count: 43
=======================
   USING BRUTE FORCE
=======================
Shortest Distance using Brute Force is 5.25 with points (1.86,64.04) and (4.26,68.71)
Euclidean Distance operations count: 121
=======================
        RUNTIME
=======================
Divide and Conquer Runtime: 0.0 ms
Bruteforce Runtime: 0.0 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.9: Masukan dan Luaran untuk 16 titik di Ruang Dimensi 2

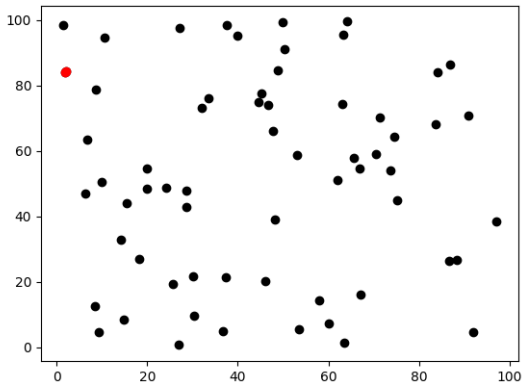


Gambar 2.10: Visualisasi Hasil untuk 16 titik di Ruang Dimensi 2

64 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
=======================
 CLOSEST PAIR OF POINTS
=======================
Masukkan jumlah titik: 64
Masukkan dimensi titik: 2
=======================
        RESULT
=======================
USING DIVIDE AND CONQUER
=======================
Shortest Distance is 0.46 with points (1.78,84.12) and (2.17,84.36)
Euclidean Distance operations count: 236
=======================
   USING BRUTE FORCE
=======================
Shortest Distance using Brute Force is 0.46 with points (1.78,84.12) and (2.17,84.36)
Euclidean Distance operations count: 2017
=======================
        RUNTIME
=======================
Divide and Conquer Runtime: 0.0 ms
Bruteforce Runtime: 1.08 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.11: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 2

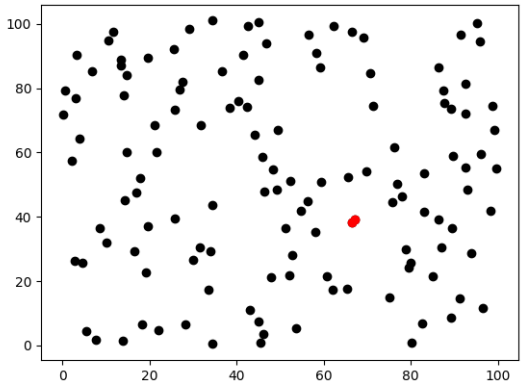


Gambar 2.12: Visualisasi Hasil untuk 64 titik di Ruang Dimensi 2

128 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
=======================
 CLOSEST PAIR OF POINTS
=======================
Masukkan jumlah titik: 128
Masukkan dimensi titik: 2
=======================
        RESULT
=======================
USING DIVIDE AND CONQUER
=======================
Shortest Distance is 1.13 with points (66.47,38.2) and (67.07,39.16)
Euclidean Distance operations count: 796
=======================
   USING BRUTE FORCE
=======================
Shortest Distance using Brute Force is 1.13 with points (66.47,38.2) and (67.07,39.16)
Euclidean Distance operations count: 8129
=======================
        RUNTIME
=======================
Divide and Conquer Runtime: 1.08 ms
Bruteforce Runtime: 4.09 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.13: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 2

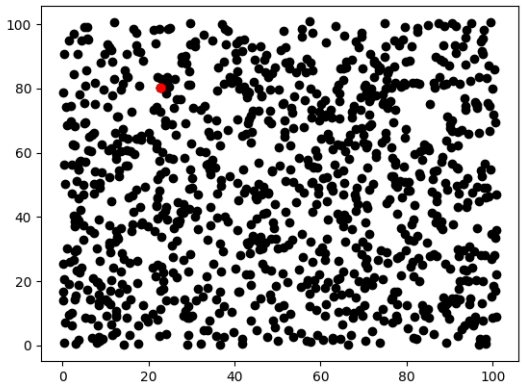


Gambar 2.14: Visualisasi Hasil untuk 128 titik di Ruang Dimensi 2

1000 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 1000
Masukkan dimensi titik: 2
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 0.01 with points (22.87,80.18) and (22.88,80.17)
Euclidean Distance operations count: 11208
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 0.01 with points (22.87,80.18) and (22.88,80.17)
Euclidean Distance operations count: 499501
========================
        RUNTIME
========================
Divide and Conquer Runtime: 9.51 ms
Bruteforce Runtime: 190.09 ms
Apakah Anda ingin melihat visualisasi pada 2D atau 3D? (y/n) y
```

Gambar 2.15: Masukan dan Luaran untuk 1000 titik di Ruang Dimensi 2



Gambar 2.16: Visualisasi Hasil untuk 1000 titik di Ruang Dimensi 2

2.3.3 Dimensi 4

16 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 16
Masukkan dimensi titik: 4
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 16.92 with points (91.75,20.86,100.09,41.1) and (77.07,28.84,98.04,42.78)
Euclidean Distance operations count: 227
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 16.92 with points (77.07,28.84,98.04,42.78) and (91.75,20.86,100.09,41.1)
Euclidean Distance operations count: 121
========================
        RUNTIME
========================
Divide and Conquer Runtime: 0.0 ms
Bruteforce Runtime: 0.0 ms
```

Gambar 2.17: Masukan dan Luaran untuk 16 titik di Ruang Dimensi 4

64 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 64
Masukkan dimensi titik: 4
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 12.71 with points (7.86,40.63,26.88,40.37) and (18.71,37.02,24.46,45.35)
Euclidean Distance operations count: 3240
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 12.71 with points (7.86,40.63,26.88,40.37) and (18.71,37.02,24.46,45.35)
Euclidean Distance operations count: 2017
========================
        RUNTIME
========================
Divide and Conquer Runtime: 2.98 ms
Bruteforce Runtime: 1.09 ms
```

Gambar 2.18: Masukan dan Luaran untuk 64 titik di Ruang Dimensi 4

128 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 128
Masukkan dimensi titik: 4
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 7.12 with points (1.51,45.7,3.13,52.17) and (5.64,43.39,0.68,56.89)
Euclidean Distance operations count: 8689
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 7.12 with points (1.51,45.7,3.13,52.17) and (5.64,43.39,0.68,56.89)
Euclidean Distance operations count: 8129
========================
        RUNTIME
========================
Divide and Conquer Runtime: 8.67 ms
Bruteforce Runtime: 4.01 ms
```

Gambar 2.19: Masukan dan Luaran untuk 128 titik di Ruang Dimensi 4

1000 Titik

```
PS C:\Users\Matthew\Documents\Institut Teknologi Bandung\Semester 4\Strategi Algoritma\Tucil2_13521007\src> python main.
py
========================
 CLOSEST PAIR OF POINTS
========================
Masukkan jumlah titik: 1000
Masukkan dimensi titik: 4
========================
        RESULT
========================
USING DIVIDE AND CONQUER
========================
Shortest Distance is 3.33 with points (97.48,14.61,100.61,64.03) and (99.73,14.43,98.65,65.48)
Euclidean Distance operations count: 314150
========================
   USING BRUTE FORCE
========================
Shortest Distance using Brute Force is 3.33 with points (97.48,14.61,100.61,64.03) and (99.73,14.43,98.65,65.48)
Euclidean Distance operations count: 499501
========================
        RUNTIME
========================
Divide and Conquer Runtime: 220.28 ms
Bruteforce Runtime: 255.34 ms
```

Gambar 2.20: 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 waktu kompilasi 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