LAPORAN

TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

MENCARI PASANGAN TITIK TERDEKAT
DENGAN ALGORITMA DIVIDE AND CONQUER

Oleh:

Hidayatullah Wildan Ghaly Buchary (13521015)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG BANDUNG

2022/2023

DAFTAR ISI

BAB I PENDAHULUAN 3
BAB II LANGKAH JALANNYA PROGRAM 4
BAB III SOURCE CODE PROGRAM 5
BAB IV INPUT DAN OUTPUT 11
BAB V KESIMPULAN 16
BAB VI SPESIFIKASI LAPTOP 17
BAB VII LAMPIRAN 18

BAB I
PENDAHULUAN

Di era digital saat ini, data berdimensi tinggi seperti informasi spasial tiga dimensi semakin banyak digunakan di berbagai bidang seperti pemodelan arsitektur, visualisasi game, pengenalan objek dan masih banyak lagi lainnya. Salah satu masalah terpenting dalam pengolahan data spasial 3D adalah menemukan pasangan titik terdekat, yaitu menemukan dua titik dalam suatu himpunan titik yang terdekat. Algoritma Divide and Conquer adalah cara yang efektif untuk memecahkan masalah ini. Dalam makalah ini, kami membahas bagaimana algoritma Divide and Conquer dapat diterapkan untuk menemukan pasangan titik terdekat secara efisien dan bagaimana efisiensi algoritma ini dibandingkan dengan pendekatan lain. Semoga artikel ini memberikan pemahaman yang lebih baik tentang algoritma Divide and Conquer dan penerapannya dalam pengolahan data spasial 3D.

Pencarian pasangan titik terdekat pada data spasial 3D dapat dilakukan dengan menggunakan algoritma brute force yaitu membandingkan setiap pasangan titik dan mencari pasangan dengan jarak terkecil. Namun, algoritma ini memiliki kompleksitas waktu yang tinggi sebesar $O(n^2)$, di mana n adalah jumlah titik dalam himpunan. Misalnya, jika kalimat berisi 1000 poin, Bruteforce akan melakukan sekitar satu juta operasi perbandingan.

Sementara itu, algoritma Divide and Conquer dapat mengurangi kompleksitas waktu menjadi O(n log n) dengan membagi himpunan titik menjadi beberapa bagian, menyelesaikan setiap bagian secara terpisah, dan kemudian menggabungkan hasilnya. Dengan cara ini, jumlah operasi perbandingan dapat dikurangi secara signifikan, memungkinkan algoritma Divide and Conquer memberikan solusi lebih cepat untuk skor besar. Selain itu, algoritma Divide and Conquer juga lebih mudah diimplementasikan dan konsepnya lebih mudah dipahami dibandingkan dengan brute force. Hal ini menjadikan algoritma ini pilihan yang lebih baik untuk masalah menemukan pasangan titik terdekat dalam data spasial 3D.

Secara umum, algoritma Divide and Conquer merupakan cara yang lebih efektif dan efisien untuk menemukan pasangan titik terdekat dalam data spasial 3D dibandingkan dengan metode brute force.

BAB II
LANGKAH JALANNYA PROGRAM

Berikut adalah langkah-langkah jalannya program yang telah dibuat:

1. Program akan dimulai dengan meminta input dari user berupa jumlah titik, dimensi, nilai maksimum titik, dan nilai minimum titik. Input dari user akan diulang selama input masih tidak sesuai dengan kriteria.
2. Program akan memanggil fungsi randomizePoints untuk melakukan pembangkitan titik random sesuai dengan input pengguna sebelumnya.
3. Program akan mulai menyimpan waktu eksekusi dan memanggil fungsi bruteforce yang akan mencari jarak terdekat antara titik yang telah dibangkitkan. Pencarian dengan bruteforce ini dilakukan dengan cara membandingkan semua titik dengan titik lainnya secara keseluruhan tanpa kecuali. Setelah itu program akan menampilkan hasil berupa jarak, titik pertama, titik kedua, total operasi euclidian, dan waktu eksekusi untuk algoritma bruteforce.
4. Program akan mulai menyimpan waktu eksekusi dan memanggil fungsi divideAndConquer yang akan mencari jarak terdekat antara titik yang telah dibangkitkan. Pencarian akan dilakukan dengan cara membagi list titik-titik yang telah dibangkitkan menjadi dua dan akan terus dilakukan hingga titik tersisa tiga atau dua. Setelah itu program akan menghitung euclidian distance dari titik-titik tersebut dan membandingkannya dengan jarak yang akan didapatkan dengan cara serupa tetapi di sisi lainnya. Setekah itu program juga akan melakukan pengecekan titik-titik yang dekat dengan perbatasan dengan membandingkannya berdasarkan jarak yang telah di dapat. Algoritma ini memanfaatkan rekursif sampai nantinya didapatkan titik-titik terdekat setelah proses rekursif berakhir. Setelah itu program akan menampilkan jarak, titik pertama, titik kedua, total operasi euclidian, dan waktu eksekusinya.
5. Program akan melakukan pengecekan terhadap dimensi yang dimasukkan pengguna. Apabila dimensi masukan pengguna paling banyak tiga maka program akan memanggil fungsi plot untuk melakukan plotting terhadap titik-titik yang telah dibangkitkan dan juga akan memberikan warna berbeda terhadap dua titik terdekat. Jika dimensi yang dimasukkan pengguna lebih dari tiga maka program akan menampilkan pesan keluaran tidak mungkin untuk melakukan plotting terhadap titik-titik yang telah dibangkitkan.

BAB III
SOURCE CODE PROGRAM

Berikut ini adalah source code dari program yang telah dibuat dengan bahasa pemrograman python:

1. Source code dalam file bruteForce.py

```
# Nama file     : bruteForce.py
# Fungsi        : Mencari titik terdekat dengan algoritma brute force

import distance as dist

# Pencarian titik terdekat dengan algoritma brute force
def bruteForce(points):
    minDistance = float('inf')
    closestPair = None
    totalOperation = 0;
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            distance, totalOperation = dist.getDistance(points[i], points[j],
totalOperation)
            if distance < minDistance:
                minDistance = distance
                closestPair = (points[i], points[j])
    return (minDistance,
            closestPair,
            totalOperation)
```

2. Source code dalam file distance.py

```
# Nama file     : distance.py
# Fungsi        : Menghitung jarak antara dua titik

import math

# Calculate the distance between two points
def getDistance(point1, point2, totalOperation):
    distance = 0
    for i in range (len(point1)):
        distance += (point1[i] - point2[i])**2
    return math.sqrt(distance), totalOperation + 1
```

3. Source code dalam file plot.py

```
# Nama file     : plot.py
# Fungsi        : Membuat plot dari titik-titik yang ada

import matplotlib.pyplot as plt

# Plot the points and the line between p1 and p2
def plot(points, p1, p2):
    x = []
    y = []
    z = []
    for point in points:
        if (point != p1 and point != p2):
            x.append(point[0])
            if (len(point) > 1):
                y.append(point[1])
                if (len(point) == 3):
                    z.append(point[2])
    if (len(point) == 3):
        ax = plt.axes(projection ='3d')
        ax.scatter(x, y, z, color='black')
        ax.set_title('3D Plot')
        plt.plot([p1[0], p2[0]], [p1[1], p2[1]], [p1[2], p2[2]], color='red')
        ax.scatter(p1[0], p1[1], p1[2], color='red')
        ax.scatter(p2[0], p2[1], p2[2], color='red')
    elif (len(point) == 2):
        plt.scatter(x, y, color='black')
        plt.plot([p1[0], p2[0]], [p1[1], p2[1]], color='red')
        plt.scatter(p1[0], p1[1], color='red')
        plt.scatter(p2[0], p2[1], color='red')
    else: # len(point) == 1
        plt.scatter(x, [0 for i in x], color = 'black')
        plt.scatter(p1[0], 0, color = 'red')
        plt.scatter(p2[0], 0, color = 'red')
        plt.plot([p1[0], p2[0]], [0, 0], color = 'red')
    plt.show()
```

4. Source code dalam file randomize.py

```
# Nama file     : randomize.py
# Fungsi        : Membangkitkan titik-titik secara random sesuai input
```

```
import random as rand

# Generate random points
def randomizePoints(totalPoints, dimension, maximum, minimum):
    points = []
    for i in range(totalPoints):
        point = []
        for j in range(dimension):
            point.append(rand.randint(minimum, maximum))
        points.append(point)
    return points
```

5. Source code dalam file divideAndConquer.py

```
# Nama file     : divideAbdConquer.py
# Fungsi        : Mencari titik terdekat dengan algoritma divide and conquer

import distance as dist

# Pencarian titik terdekat dengan algoritma divide and conquer
def divideAndConquer(points, totalOperation):
    if (len(points) == 2):
        distance, totalOperation = dist.getDistance(points[0], points[1],
totalOperation)
        return (distance, points[0], points[1], totalOperation)

    elif (len(points) == 3):
        minDistance = float('inf')
        point1 = points[0]
        point2 = points[1]
        for i in range(len(points)):
            for j in range(i+1, len(points)):
                dis, totalOperation = dist.getDistance(points[i], points[j],
totalOperation)
                if (dis < minDistance):
                    minDistance = dis
                    point1 = points[i]
                    point2 = points[j]
        return (minDistance,
                point1, point2,
                totalOperation)

    else:
        mid = (len(points) // 2)
        leftPointsDist, leftPoint1, leftPoint2, totalOperation =
divideAndConquer(points[:mid], totalOperation)
```

```
        rightPointsDist, rightPoint1, rightPoint2, totalOperation =
divideAndConquer(points[mid:], totalOperation)
        if (leftPointsDist < rightPointsDist):
            minDistance = leftPointsDist
            point1 = leftPoint1
            point2 = leftPoint2
        else:
            minDistance = rightPointsDist
            point1 = rightPoint1
            point2 = rightPoint2

        midX = points[mid][0]
        minPoints = []
        for point in points:
            if (abs(point[0] - midX) < minDistance):
                minPoints.append(point)

        for i in range(len(minPoints)):
            for j in range(i+1, len(minPoints)):
                dis, totalOperation = dist.getDistance(minPoints[i],
minPoints[j], totalOperation)
                if (dis < minDistance):
                    minDistance = dis
                    point1 = minPoints[i]
                    point2 = minPoints[j]

        return (minDistance,
                point1, point2,
                totalOperation)
```

6. Source code dalam file main.py

```
# Nama file     : main.py
# Fungsi        : Menjadi main dari program

import bruteForce as bf
import divideAndConquer as dc
import randomize as rand
import time
import plot

# Main program
def main():
    # Input
    while (True):
        print("\n=======================================")
```

```
        print("           Closest Pair Problem          ")
        print("=========================================")
        jumlahTitik = int(input ("Enter number of points    : "))
        dimensi = int(input     ("Enter dimension           : "))
        maxim = int(input       ("Enter maximum value       : "))
        minim = int(input       ("Enter minimum value       : "))
        if (maxim < minim):
            print("Maximum value must be greater than minimum value")
        elif (jumlahTitik < 2):
            print("Number of points must be greater than 1")
        elif (dimensi < 1):
            print ("Dimension must be greater than 0")
        else:
            print("Generating points...")
            break

    # Generate random points
    points = rand.randomizePoints(jumlahTitik, dimensi, maxim, minim)
    points.sort(key=lambda x: x[0])

    # Brute Force
    start = time.time()
    dist, (p1, p2), totalOperation = bf.bruteForce(points)
    end = time.time()

    # Print result
    print("\nBrute Force: ")
    print("     Distance            : ", round(dist,3))
    print("     Point 1             : ", p1)
    print("     Point 2             : ", p2)
    print("     Total Operation     : ", totalOperation)
    print("     Time                : ", round((end - start)*1000,4), "ms")

    # Divide and Conquer
    start = time.time()
    dist, p1, p2, totalOperation = dc.divideAndConquer(points, 0)
    end = time.time()

    # Print result
    print("\nDivide and Conquer : ")
    print("     Distance            : ", round(dist,3))
    print("     Point 1             : ", p1)
    print("     Point 2             : ", p2)
    print("     Total Operation     : ", totalOperation)
    print("     Time                : ", round((end - start)*1000,4), "ms")

    if (dimensi <= 3):
        plot.plot(points, p1, p2)
```

```
    else:
        print("\nPlotting is not available for dimension greater than 3")

if __name__ == '__main__':
    main()
```

BAB IV
INPUT DAN OUTPUT

1. Error handling input number of points

```
========================================
          Closest Pair Problem
========================================
Enter number of points    : -1
Enter dimension           : 3
Enter maximum value       : 100
Enter minimum value       : -100
Number of points must be greater than 1
```

2. Error handling dimension

```
========================================
          Closest Pair Problem
========================================
Enter number of points    : 100
Enter dimension           : -1
Enter maximum value       : 100
Enter minimum value       : -100
Dimension must be greater than 0
```

3. Error handling maximum and minimum value

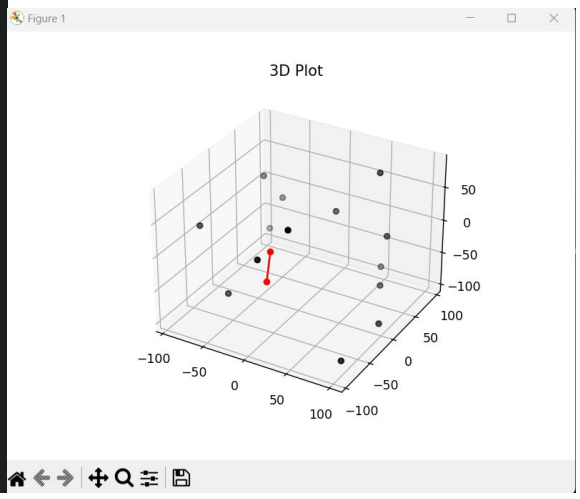
```
========================================
          Closest Pair Problem
========================================
Enter number of points    : 100
Enter dimension           : 3
Enter maximum value       : -100
Enter minimum value       : 100
Maximum value must be greater than minimum value
```

4. Jumlah titik 16 dan dimensi 3

```
========================================
          Closest Pair Problem
========================================
Enter number of points    : 16
Enter dimension           : 3
Enter maximum value       : 100
Enter minimum value       : -100
Generating points...

Brute Force:
     Distance             :  37.443
     Point 1              :  [-18, -26, -9]
     Point 2              :  [-6, -53, -32]
     Total Operation      :  120
     Time                 :  0.0 ms

Divide and Conquer :
     Distance             :  37.443
     Point 1              :  [-18, -26, -9]
     Point 2              :  [-6, -53, -32]
     Total Operation      :  81
     Time                 :  0.0 ms
```

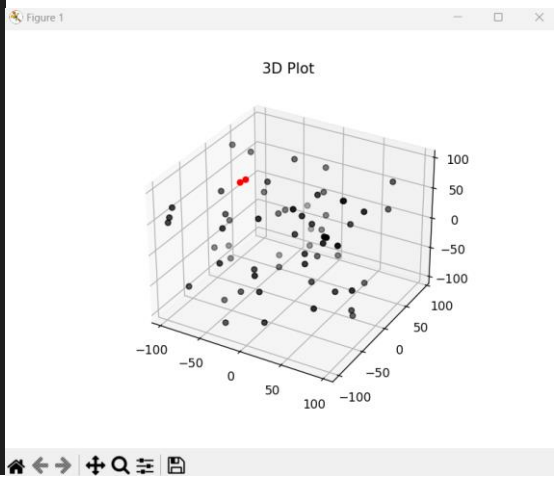


5. Jumlah titik 64 dan dimensi 3

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 64
Enter dimension            : 3
Enter maximum value        : 100
Enter minimum value        : -100
Generating points...

Brute Force:
     Distance              :  6.403
     Point 1               :  [-63, -8, 69]
     Point 2               :  [-59, -4, 72]
     Total Operation       :  2016
     Time                  :  2.3594 ms

Divide and Conquer :
     Distance              :  6.403
     Point 1               :  [-63, -8, 69]
     Point 2               :  [-59, -4, 72]
     Total Operation       :  579
     Time                  :  0.0 ms
```

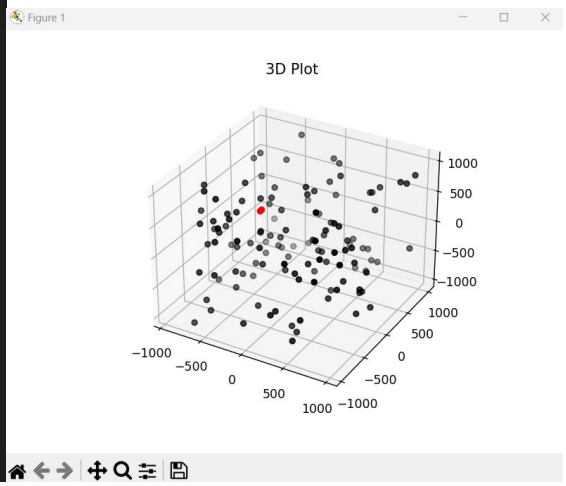


6. Jumlah titik 128 dan dimensi 3

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 128
Enter dimension            : 3
Enter maximum value        : 1000
Enter minimum value        : -1000
Generating points...

Brute Force:
     Distance              :  63.64
     Point 1               :  [-250, -242, 571]
     Point 2               :  [-235, -299, 595]
     Total Operation       :  8128
     Time                  :  7.5161 ms

Divide and Conquer :
     Distance              :  63.64
     Point 1               :  [-250, -242, 571]
     Point 2               :  [-235, -299, 595]
     Total Operation       :  2045
     Time                  :  2.0187 ms
```

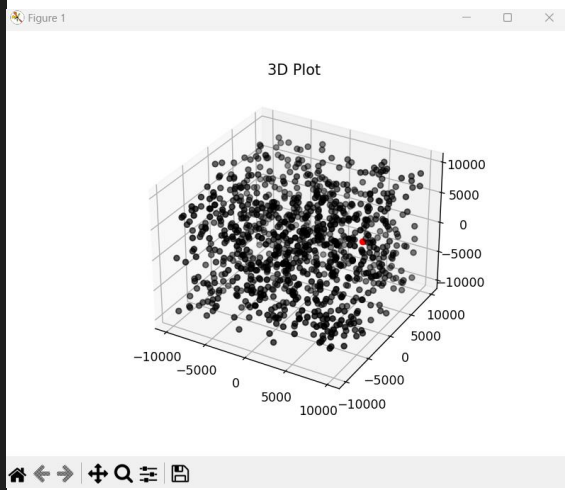


7. Jumlah titik 1000 dan dimensi 3

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 1000
Enter dimension            : 3
Enter maximum value        : 10000
Enter minimum value        : -10000
Generating points...

Brute Force:
     Distance              :  147.017
     Point 1               :  [4184, 6850, -2974]
     Point 2               :  [4277, 6932, -3053]
     Total Operation       :  499500
     Time                  :  514.4589 ms

Divide and Conquer :
     Distance              :  147.017
     Point 1               :  [4184, 6850, -2974]
     Point 2               :  [4277, 6932, -3053]
     Total Operation       :  48127
     Time                  :  51.4243 ms
```

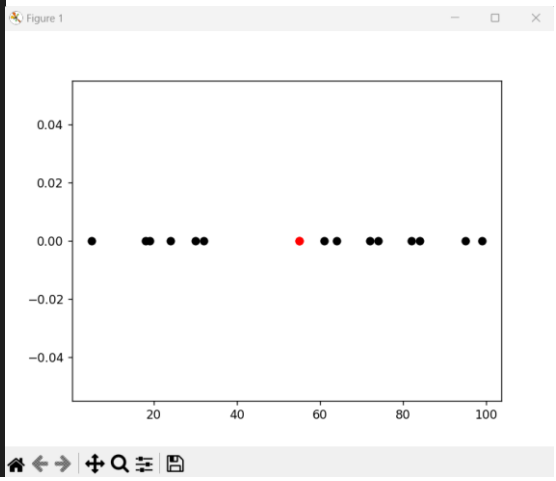


8. Jumlah titik 16 dan dimensi 1

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 16
Enter dimension            : 1
Enter maximum value        : 100
Enter minimum value        : 0
Generating points...

Brute Force:
     Distance              :  0.0
     Point 1               :  [55]
     Point 2               :  [55]
     Total Operation       :  120
     Time                  :  0.0 ms

Divide and Conquer :
     Distance              :  0.0
     Point 1               :  [55]
     Point 2               :  [55]
     Total Operation       :  9
     Time                  :  0.0 ms
```

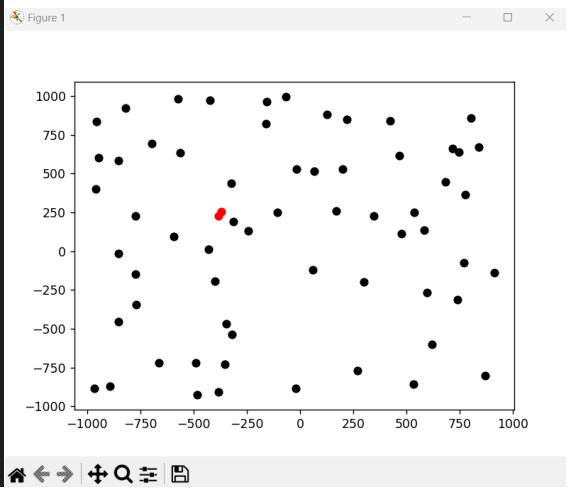


9. Jumlah titik 64 dan dimensi 2

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 64
Enter dimension            : 2
Enter maximum value        : 1000
Enter minimum value        : -1000
Generating points...

Brute Force:
     Distance              :  30.017
     Point 1               :  [-383, 226]
     Point 2               :  [-368, 252]
     Total Operation       :  2016
     Time                  :  2.1696 ms

Divide and Conquer :
     Distance              :  30.017
     Point 1               :  [-383, 226]
     Point 2               :  [-368, 252]
     Total Operation       :  434
     Time                  :  0.0 ms
```

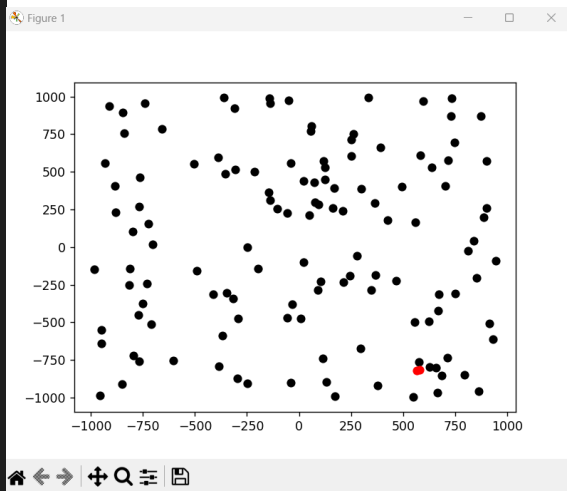


10. Jumlah titik 128 dan dimensi 2

```
========================================
          Closest Pair Problem
========================================
Enter number of points     : 128
Enter dimension            : 2
Enter maximum value        : 1000
Enter minimum value        : -1000
Generating points...

Brute Force:
     Distance              :  16.125
     Point 1               :  [567, -822]
     Point 2               :  [581, -814]
     Total Operation       :  8128
     Time                  :  6.1715 ms

Divide and Conquer :
     Distance              :  16.125
     Point 1               :  [567, -822]
     Point 2               :  [581, -814]
     Total Operation       :  1089
     Time                  :  1.0045 ms
```



11. Jumlah titik 1000 dan dimensi 4

```
========================================
          Closest Pair Problem
========================================
Enter number of points    : 1000
Enter dimension           : 4
Enter maximum value       : 10000
Enter minimum value       : -10000
Generating points...

Brute Force:
     Distance             :  496.735
     Point 1              :  [-1938, 3979, -3943, -8295]
     Point 2              :  [-1821, 3757, -4370, -8333]
     Total Operation      :  499500
     Time                 :  598.7332 ms

Divide and Conquer :
     Distance             :  496.735
     Point 1              :  [-1938, 3979, -3943, -8295]
     Point 2              :  [-1821, 3757, -4370, -8333]
     Total Operation      :  98481
     Time                 :  124.6881 ms

Plotting is not available for dimension greater than 3
```

12. Jumlah titik 1000 dan dimensi 8

```
========================================
          Closest Pair Problem
========================================
Enter number of points    : 1000
Enter dimension           : 8
Enter maximum value       : 10000
Enter minimum value       : -10000
Generating points...

Brute Force:
     Distance             :  3525.221
     Point 1              :  [-9911, -6273, 3048, -5311, 2337, -6812, -8540, 8311]
     Point 2              :  [-7488, -4844, 3363, -3734, 2052, -7606, -9640, 8231]
     Total Operation      :  499500
     Time                 :  1059.3045 ms

Divide and Conquer :
     Distance             :  3525.221
     Point 1              :  [-9911, -6273, 3048, -5311, 2337, -6812, -8540, 8311]
     Point 2              :  [-7488, -4844, 3363, -3734, 2052, -7606, -9640, 8231]
     Total Operation      :  450375
     Time                 :  956.3377 ms

Plotting is not available for dimension greater than 3
```

BAB V
KESIMPULAN

Secara umum, algoritma Divide and Conquer memecah masalah yang kompleks menjadi submasalah yang lebih sederhana dan kemudian menggabungkan solusi submasalah tersebut untuk memecahkan masalah utama.

Dalam konteks mencari pasangan titik terdekat, algoritma Divide and Conquer dapat digunakan dengan membagi himpunan titik menjadi dua bagian yang lebih kecil secara rekursif dan mencari pasangan titik terdekat di masing-masing bagian tersebut. Kemudian, pasangan titik terdekat dari kedua bagian tersebut dibandingkan untuk menentukan pasangan titik terdekat keseluruhan.

Dibandingkan dengan brute force, algoritma Divide and Conquer lebih efisien untuk mencari pasangan titik terdekat pada himpunan titik yang besar karena memiliki kompleksitas waktu yang lebih rendah. Pada brute force, setiap pasangan titik harus dibandingkan satu sama lain untuk menentukan pasangan titik terdekat. Hal ini membutuhkan waktu $O(n^2)$, di mana n adalah jumlah titik dalam himpunan. Sementara pada algoritma Divide and Conquer, kompleksitas waktu yang dibutuhkan hanya $O(n \log(n))$ yang lebih efisien daripada brute force.

BAB VI

SPESIFIKASI LAPTOP

Pengujian input dan output pada bab IV dijalankan pada spesifikasi laptop sebagai berikut:

1. Operating System : Microsoft Windows 11 Home Single Language 64-bit 10.0.22621
2. Processor : AMD Ryzen 7 5800H with Radeon Graphics
3. Graphic Card : AMD Radeon™ Graphics
4. Storage : SSD/EMMC 1 476GB, SAMSUNG MZVLQ512HBLU-00B00
5. Memory : Total 8GB

BAB VII
LAMPIRAN

Link repository github: https://github.com/WildanGhaly/Tucil2_13521015

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	✓	