Tugas Kecil 2 IF2211 Strategi Algoritma

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

Ditulis oleh:

Muhammad Rifko Favian 13521075

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023

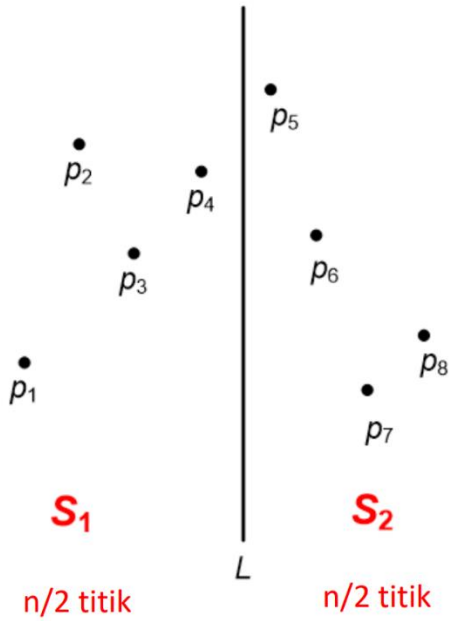
BAGIAN I
DESKRIPSI MASALAH

Mencari sepasang titik terdekat dengan Algoritma *Divide and Conquer* sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini Anda diminta mengembangkan algoritma mencari sepasang titik terdekat pada bidang 3D. Misalkan terdapat n buah titik pada ruang 3D. Setiap titik P di dalam ruang dinyatakan dengan koordinat P = (x, y, z). Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titk P_1 = (x_1, y_1, z_1) dan P_2 = (x_2, y_2, z_2) dihitung dengan rumus Euclidean berikut:

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

BAGIAN II
LANGKAH ALGORITMA

Mencari sepasang titik terdekat pada vektor tiga dimensi dengan menggunakan pendekatan algoritma *divide and conquer* diawali dengan membagi semua titik-titik pada vector menjadi dua bagian.



Setelah di-*divide*, *conquer* pembagian tersebut dengan keadaan 3 kasus: pencarian pasangan titik terdekat pada S1, pencarian pasangan titik terdekat pada S2, pencarian pasangan titik terdekat pada titik-titik yang terpisah. Pencarian pasangan pada S1 dan S2 dilakukan dengan men-*divide* lagi titik-titik pada himpunan sampai tersisa dua titik (kasus genap) atau tiga titik (kasus ganjil), kemudian di-*conquer* dengan menghitung dan mencari (apabila tiga) pasangan terdekat menggunakan rumus *euclidean*. Lakukan hal ini sampai ditemukan pasangan titik terdekat pada S1 dan S2 kemudian tentukan nilai mana yang lebih kecil antara kedua pembagian tersebut. Untuk kasus yang ketiga, yaitu pencarian titik terdekat pada titik-titik yang terpisah, dilakukan dengan menentukan titik-titik yang dekat dengan garis pembagi, yaitu titik-titik yang berjarak *d* dari garis di mana *d* adalah nilai jarak terkecil dari pencarian dua kasus sebelumnya. Kemudian titik-titik bagian kiri dan bagian kanan masing-masing dihitung jaraknya dengan rumus *euclidean* untuk menentukan apakah terdapat nilai yang lebih kecil dibanding nilai *d*.

Apabila pencarian titik terdekat dilakukan dengan pendekatan algoritma *brute force*, maka langkahnya cukup menghitung semua kemungkinan jarak sepasang titik kemudian cari nilai yang terkecil di antara nilai-nilai tersebut.

BAGIAN III
PROGRAM SOURCE

ClosestPair.py

```
import math
import random

euclideanCalled = 0

def randomPoint(n, dim):
    arrayPoint = []
    for i in range(0, n):
        point = []
        for j in range(0, dim):
            sumbu = round(random.uniform(-1000, 1000), 2)
            point.append(sumbu)
        arrayPoint.append(point)
    return arrayPoint


def euclideanDistance(point1, point2):
    distance = float(0)
    for i in range(0, len(point1)):
        distance += math.pow((point1[i]-point2[i]), 2)
    global euclideanCalled
    euclideanCalled += 1
    return math.sqrt(distance)


def sortPointbyX(arrayPoint):
    for i in range(0, len(arrayPoint)):
        for j in range(0, len(arrayPoint)-1-i):
            if arrayPoint[j][0] > arrayPoint[j+1][0]:
                arrayPoint[j], arrayPoint[j+1] = arrayPoint[j+1], arrayPoint[j]
    return arrayPoint


def closestPairDivideNConquer(arrayPoint, n, dim):
    if (n == 2):
        distance = euclideanDistance(arrayPoint[0], arrayPoint[1])
        result = [distance, arrayPoint[0], arrayPoint[1]]
        return result
    elif (n == 3):
        d1 = euclideanDistance(arrayPoint[0], arrayPoint[1])
        d2 = euclideanDistance(arrayPoint[0], arrayPoint[2])
        d3 = euclideanDistance(arrayPoint[1], arrayPoint[2])
        if (d1 <= d2) and (d1 <= d3):
            result = [d1, arrayPoint[0], arrayPoint[1]]
        elif (d2 <= d1) and (d2 <= d3):
            result = [d2, arrayPoint[0], arrayPoint[2]]
        else:
            result = [d3, arrayPoint[1], arrayPoint[2]]
        return result
    else:
        mid = n//2
```

```
        left = arrayPoint[:mid]
        right = arrayPoint[mid:]

        dleft = closestPairDivideNConquer(left, mid, dim)
        dright = closestPairDivideNConquer(right, n-mid, dim)

        result = []

        if (dleft[0] < dright[0]):
            result = dleft
        else:
            result = dright

        strip = []

        if (n % 2 == 0):
            midPoint = (arrayPoint[mid][0]+arrayPoint[mid+1][0])/2
        else:
            midPoint = arrayPoint[mid][0]

        for i in range(0, n):
            if (arrayPoint[i][0] >= (midPoint - result[0])) and (arrayPoint[i][0]
<= (midPoint + result[0])):
                strip.append(arrayPoint[i])

        if (len(strip) > 1):
            for i in range(0, len(strip)):
                for j in range(i+1, len(strip)):
                    found = True
                    for k in range(dim):
                        if found:
                            found = found and (
                                abs(strip[i][k]-strip[j][k]) < result[0])
                    if (found):
                        distance = euclideanDistance(strip[i], strip[j])
                        if (distance < result[0]):
                            result = [distance, strip[i], strip[j]]

        return result


def closestPairBruteForce(arrayPoint, n):
    distance = euclideanDistance(arrayPoint[0], arrayPoint[1])
    result = [distance, arrayPoint[0], arrayPoint[1]]

    for i in range(0, n):
        for j in range(i+1, n):
            temp = euclideanDistance(arrayPoint[i], arrayPoint[j])
            if (temp < distance):
                distance = temp
                result = [distance, arrayPoint[i], arrayPoint[j]]

    return result
```

main.py

```
import ClosestPair as util
import time
import platform

dim = int(input("Masukkan banyak dimensi : "))

while dim < 2:
    print("Masukan tidak valid. Dibutuhkan minimal 2 dimensi\n")
    dim = int(input("Masukkan banyak dimensi : "))

n = int(input("Masukkan banyak titik : "))

while n < 2:
    print("Masukan tidak valid. Dibutuhkan minimal 2 titik untuk menghitung
jarak\n")
    n = int(input("Masukkan banyak titik : "))

arrayPoint = util.randomPoint(n, dim)
arrayPoint = util.sortPointbyX(arrayPoint)

startDnC = time.time()
resultDnC = util.closestPairDivideNConquer(arrayPoint, n, dim)
timeDnC = time.time() - startDnC
euclideanDnC = util.euclideanCalled

startBF = time.time()
resultBF = util.closestPairBruteForce(arrayPoint, n)
timeBF = time.time() - startBF
euclideanBF = util.euclideanCalled - euclideanDnC


print("\n=================== DIVIDE AND CONQUER ====================")
print("Point 1                 :", resultDnC[1])
print("Point 2                 :", resultDnC[2])
print("Distance between them   :", resultDnC[0])
print("Execution time          :", timeDnC, "s")
print("Euclidean function is called", str(euclideanDnC) + "x\n")


print("====================== BRUTE FORCE ======================")
print("Point 1                 :", resultBF[1])
print("Point 2                 :", resultBF[2])
print("Distance between them   :", resultBF[0])
print("Execution time          :", timeBF, "s")
print("Euclidean function is called", str(euclideanBF) + "x\n")

print("Run at", platform.platform(), platform.processor(), platform.machine())
```

BAGIAN IV
CONTOH PROGRAM

1. Masukan n = 16

```
Masukkan banyak dimensi : 3
Masukkan banyak titik : 16

================== DIVIDE AND CONQUER ==================
Point 1                : [-810.29, -93.96, 957.96]
Point 2                : [-620.97, 45.4, 988.51]
Distance between them  : 237.0581669126798
Execution time         : 0.0 s
Euclidean function is called 17x

====================== BRUTE FORCE =====================
Point 1                : [-810.29, -93.96, 957.96]
Point 2                : [-620.97, 45.4, 988.51]
Distance between them  : 237.0581669126798
Execution time         : 0.0 s
Euclidean function is called 121x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

2. Masukan n = 64

```
Masukkan banyak dimensi : 3
Masukkan banyak titik : 64

================== DIVIDE AND CONQUER ==================
Point 1                : [274.17, 241.67, 271.18]
Point 2                : [301.26, 289.23, 282.26]
Distance between them  : 55.84432021253371
Execution time         : 0.001051187515258789 s
Euclidean function is called 72x

====================== BRUTE FORCE =====================
Point 1                : [274.17, 241.67, 271.18]
Point 2                : [301.26, 289.23, 282.26]
Distance between them  : 55.84432021253371
Execution time         : 0.0019638538360595703 s
Euclidean function is called 2017x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

3. Masukan n = 128

```
Masukkan banyak dimensi : 3
Masukkan banyak titik : 128

================== DIVIDE AND CONQUER ==================
Point 1                : [-591.66, 989.65, -607.06]
Point 2                : [-548.45, 978.42, -597.86]
Distance between them  : 45.58351675770522
Execution time         : 0.0009684562683105469 s
Euclidean function is called 165x

====================== BRUTE FORCE =======================
Point 1                : [-591.66, 989.65, -607.06]
Point 2                : [-548.45, 978.42, -597.86]
Distance between them  : 45.58351675770522
Execution time         : 0.0081024169921875 s
Euclidean function is called 8129x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

4. Masukan n = 1000

```
Masukkan banyak dimensi : 3
Masukkan banyak titik : 1000

================== DIVIDE AND CONQUER ==================
Point 1                : [964.18, 910.63, -279.38]
Point 2                : [971.48, 903.89, -274.56]
Distance between them  : 11.043097391583624
Execution time         : 0.020740270614624023 s
Euclidean function is called 1381x

====================== BRUTE FORCE =======================
Point 1                : [964.18, 910.63, -279.38]
Point 2                : [971.48, 903.89, -274.56]
Distance between them  : 11.043097391583624
Execution time         : 0.4183840751647949 s
Euclidean function is called 499501x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

5. Masukan dimensi 2

```
Masukkan banyak dimensi : 2
Masukkan banyak titik : 2211

================== DIVIDE AND CONQUER ==================
Point 1                : [-87.32, 68.74]
Point 2                : [-87.2, 68.99]
Distance between them  : 0.27730849247723677
Execution time         : 0.019987106323242188 s
Euclidean function is called 2690x

====================== BRUTE FORCE =======================
Point 1                : [-87.32, 68.74]
Point 2                : [-87.2, 68.99]
Distance between them  : 0.27730849247723677
Execution time         : 1.6625995635986328 s
Euclidean function is called 2443156x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

6. Masukan dimensi > 3

```
Masukkan banyak dimensi : 13
Masukkan banyak titik : 521

================== DIVIDE AND CONQUER ==================
Point 1               : [-449.33, 335.18, -354.61, 681.92, 397.88, -330.31, 541.5, -868.31, 177.98, 440.69, -561.2, 171.28, -715.04]
Point 2               : [-224.41, -16.03, -630.41, 785.46, 463.22, -492.12, 516.28, -338.33, 254.01, 740.78, -418.61, 33.59, -387.64]
Distance between them  : 902.7768582545744
Execution time         : 0.23735356330871582 s
Euclidean function is called 17036x

====================== BRUTE FORCE =====================
Point 1               : [-449.33, 335.18, -354.61, 681.92, 397.88, -330.31, 541.5, -868.31, 177.98, 440.69, -561.2, 171.28, -715.04]
Point 2               : [-224.41, -16.03, -630.41, 785.46, 463.22, -492.12, 516.28, -338.33, 254.01, 740.78, -418.61, 33.59, -387.64]
Distance between them  : 902.7768582545744
Execution time         : 0.31998443603515625 s
Euclidean function is called 135461x

Run at Windows-10-10.0.22621-SP0 Intel64 Family 6 Model 140 Stepping 1, GenuineIntel AMD64
```

BAGIAN V
LINK REPOSITORY

Link repository GitHub : https://github.com/Mifkiyan/Tucil2_13521075

BAGIAN VI
CHECK LIST

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