LAPORAN TUGAS KECIL 2 IF2211

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Semester II Tahun Akademik 2022/2023

Disusun oleh:

Zidane Firzatullah **13521163**

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

BAB I
ALGORITMA

I. Pendekatan Brute Force

1. Loop (outer loop) seluruh elemen list berisi point
2. Buat loop di dalam outer loop untuk mengiterasi seluruh elemen list berisi point (nested)
3. Jika pasangan terdekat belum memiliki nilai atau pasangan yang diobservasi memiliki jarak yang lebih kecil dari pasangan terdekat, simpan pasangan tersebut sebagai pasangan terdekat

II. Pendekatan Divide and Conquer

Kondisi: list of points sudah terurut berdasarkan nilai sumbu x.

1. Jika ukuran list lebih kecil atau sama dengan 3, pecahkan dengan metode brute force.
2. Bagi list menjadi 2 bagian, list pertama elemen 0-mid dan list kedua elemen mid-akhir dengan mid = size//2.
3. Lakukan algoritma divide and conquer terhadap list yang sudah dibagi menjadi 2 bagian.
4. Ambil jarak terkecil dari langkah 3.
5. Ambil array baru yang terdiri dari elemen-elemen dengan perbedaan nilai titik x elemen dengan nilai titik x elemen array dengan index mid kurang dari jarak terkecil.
6. Lakukan perbandingan jarak untuk pasangan pada elemen array baru (antara titik di sebelah kiri elemen index mid dengan sebelah kanan elemen). Jika perbedaan nilai salah satu sumbu lebih besar dari jarak terkecil pada langkah 3, maka proses untuk pasangan tersebut dilewati. Jika jarak dari pasangan tersebut lebih kecil dari jarak terkecil pada langkah 3, maka simpan sebagai jarak terkecil.

BAB II
SOURCE CODE

File: main.py

```
import time

import util
from point import Point
import solver


def start_io():
    print("masukkan data dengan format: dimensi<spasi>banyak_titik")

    inp: list[str]
    dim: int
    num: int

    while True:
        inp = input(">> ").split(" ")

        if len(inp) != 2:
            print("format input salah")
            continue

        try:
            dim = int(inp[0])
            num = int(inp[1])
        except ValueError:
            print("format input salah")
            continue

        if dim < 2 or num < 2:
            print("input tidak valid. batasan: dim >= 2 dan num >=
2")
            continue

        break

    start = time.time_ns()
    points = util.sort(util.generate_points(3, 20))
    Point.euclidean_count = 0
    res = solver.get_closest_points(points, points)
    end = time.time_ns()

    print("pasang titik terdekat: \n", res[1][0], '\n', res[1][1])
    print("banyak operasi euclidean distance:",
```

```
Point.euclidean_count)
    print("waktu (points generation, sort, main dnc):", float(end -
start) / 1000000000, "s")
    print("spesifikasi komputer: Intel Core i7-10750H, Intel UHD")

    if dim == 3:
        vis = input("tampilkan visualisasi 3d? (y/any): ")
        if vis.lower() == 'y':
            util.visualise(points, res[1])


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

File: point.py

```
from __future__ import annotations
from math import sqrt

euclidean_count: int = 0


class Point:
    euclidean_count: int = 0

    def __init__(self, dim, points):
        self.dim = dim
        self.points = points

    def get_distance_to(self, point2: Point) -> float:
        if self.dim != point2.dim:
            raise Exception("dimensions do not match")

        Point.euclidean_count += 1

        sum: float = 0
        for i in range(self.dim):
            sum += pow(self.points[i] - point2.points[i], 2)

        return sqrt(sum)

    def equals(self, point2: Point) -> bool:
        if self.dim != point2.dim:
            raise Exception("dimensions do not match")

        for i in range(self.dim):
            if not self.points[i] == point2.points[i]:
                return False
        return True

    def __repr__(self):
```

```
        strs = ''
        for i in range(self.dim):
            strs += 'e' + str(i + 1) + ' : ' + str(self.points[i]) +
' '

        return strs
```

File: solver.py

```
from point import Point
import util


def __get_brute_force__(points: list[Point]) -> tuple[float,
tuple[Point, Point]]:
    closest: tuple[float, tuple[Point, Point]] = None
    size = len(points)

    for i in range(size):
        for j in range(size):
            if i == j:
                continue
            dist = points[i].get_distance_to(points[j])
            if closest is None or dist < closest[0]:
                closest = dist, (points[i], points[j])

    return closest


def get_closest_points(points: list[Point], intact_p: list[Point])
-> tuple[float, tuple[Point, Point]]:
    size = len(points)

    if size <= 3:
        return __get_brute_force__(points)

    mid = int(size / 2)
    left_arr, right_arr = points[:mid], points[mid:]
    left_closest = get_closest_points(left_arr, intact_p)
    right_closest = get_closest_points(right_arr, intact_p)

    closest = left_closest if left_closest[0] < right_closest[0]
else right_closest

    mid_point = points[mid]
    left_arr = list(filter(lambda point: abs(point.points[0] -
mid_point.points[0]) <= closest[0], left_arr))
    right_arr = list(filter(lambda point: abs(point.points[0] -
mid_point.points[0]) <= closest[0], right_arr))

    for left_el in left_arr:
```

```
        for right_el in right_arr:
            if left_el.dim != right_el.dim:
                raise Exception("fatal error")
            found_bigger = False
            for j in range(left_el.dim):
                if abs(left_el.points[j] - right_el.points[j]) >
closest[0]:
                    found_bigger = True
                    break
            if found_bigger:
                continue
            dist = left_el.get_distance_to(right_el)
            if dist < closest[0]:
                closest = dist, (left_el, right_el)

    return closest
```

File: util.py

```
import random
import matplotlib.pyplot as plt

from point import Point


def generate_points(dim: int, length: int) -> list[Point]:
    points: list[Point] = []

    for i in range(length):
        # generate bounded float point number
        points.append(Point(dim, [round(random.uniform(-2000.0,
2000.0), 1) for _ in range(dim)]))

    return points


def visualise(points: list[Point], closest_pair: tuple[Point,
Point]):
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    for point in points:
        if not (point.equals(closest_pair[0]) or
point.equals(closest_pair[1])):
            ax.scatter(point.points[0], point.points[1],
point.points[2], c=['#d62728'])  # red
        else:
            ax.scatter(point.points[0], point.points[1],
point.points[2], c=['#9467bd'])  # purple
```

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

    plt.show()


def sort(points: list[Point]) -> list[Point]:
    size = len(points)
    if size == 1:
        return points

    mid = size//2
    left_arr = sort(points[:mid])
    right_arr = sort(points[mid:])

    l_idx = r_idx = m_idx = 0

    while l_idx < len(left_arr) and r_idx < len(right_arr):
        if left_arr[l_idx].points[0] > right_arr[r_idx].points[0]:
            points[m_idx] = right_arr[r_idx]
            r_idx += 1
        else:
            points[m_idx] = left_arr[l_idx]
            l_idx += 1
        m_idx += 1

    while l_idx < len(left_arr):
        points[m_idx] = left_arr[l_idx]
        l_idx += 1
        m_idx += 1

    while r_idx < len(right_arr):
        points[m_idx] = right_arr[r_idx]
        r_idx += 1
        m_idx += 1

    return points
```

BAB III
CONTOH MASUKAN DAN LUARAN

n=16

```
masukkan data dengan format: dimensi<spasi>banyak_titik
>> 3 16
pasang titik terdekat:
 e1 : 111.0 e2 : 554.3 e3 : -727.7
 e1 : 254.4 e2 : 182.1 e3 : -369.6
banyak operasi euclidean distance: 20
waktu (points generation, sort, main dnc): 0.000758759 s
spesifikasi komputer: Intel Core i7-10750H, Intel UHD
tampilkan visualisasi 3d? (y/any): n
```

n = 64

```
masukkan data dengan format: dimensi<spasi>banyak_titik
>> 3 64
pasang titik terdekat:
 e1 : 748.3 e2 : -1693.4 e3 : 1470.9
 e1 : 773.1 e2 : -1462.7 e3 : 1539.9
banyak operasi euclidean distance: 85
waktu (points generation, sort, main dnc): 0.001616445 s
spesifikasi komputer: Intel Core i7-10750H, Intel UHD
tampilkan visualisasi 3d? (y/any): n
```

n=128

```
masukkan data dengan format: dimensi<spasi>banyak_titik
>> 3 128
pasang titik terdekat:
 e1 : -1856.3 e2 : -659.2 e3 : -1079.1
 e1 : -1797.7 e2 : -491.3 e3 : -1034.0
banyak operasi euclidean distance: 175
waktu (points generation, sort, main dnc): 0.008061942 s
spesifikasi komputer: Intel Core i7-10750H, Intel UHD
tampilkan visualisasi 3d? (y/any): n
```

n=1000

```
masukkan data dengan format: dimensi<spasi>banyak_titik
>> 3 1000
pasang titik terdekat:
 e1 : -1179.5 e2 : 472.5 e3 : 935.8
 e1 : -1179.5 e2 : 451.0 e3 : 910.4
banyak operasi euclidean distance: 1542
waktu (points generation, sort, main dnc): 0.02376994 s
spesifikasi komputer: Intel Core i7-10750H, Intel UHD
tampilkan visualisasi 3d? (y/any): n
```

LAMPIRAN

1. Pranala repository github
   https://github.com/zidane-itb/Tucil2_13521163

2. Tabel Pengerjaan

Point	Ya	Tidak
Program berhasil dikompilasi tanpa ada kesalahan.	Y	
Program berhasil running.	Y	
Program dapat menerima masukan dan menuliskan luaran.	Y	
Luaran program sudah benar (solusi closest pair benar)	Y	
Bonus 1 dikerjakan	Y	
Bonus 2 dikerjakan	Y	