Информатика. Задания на кластеризацию. Повторение материала.

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике, каждая из которых находится от хотя бы одной другой звезды на расстоянии не более R условных единиц. Каждая звезда обязательно принадлежит только одному из кластеров.

Тройная звездная система – это система, в которой три звезды попарно находятся на расстоянии не более t. При этом других звезд на расстоянии менее t у этих трех звезд быть не должно.

Под расстоянием понимается расстояние Евклида между двумя точками $A(x_1, y_1)$ и $B(x_2, y_2)$ на плоскости, которое вычисляется по формуле:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Аномалиями назовём точки, находящиеся на расстоянии более одной условной единицы от точек кластеров. При расчётах аномалии учитывать не нужно.

В файле Б хранятся данные о звёздах **четырех** кластеров, где $R = 0.45$, $t = 0.01$ для каждого кластера. Известно, что количество звёзд не превышает 10 000. Структура хранения информации о звездах в файле Б аналогична файлу А.

Для каждого в каждом кластере файла найдите тройную звезду, которая образует треугольник с наименьшим периметром. Затем вычислите два числа: P_x — среднее арифметическое абсцисс найденных звезд, и P_y – среднее арифметическое ординат найденных звезд.

В ответе запишите четыре числа через пробел: сначала целую часть произведения $P_x \cdot 10000$ для файла А, затем $P_y \cdot 10000$ для файла А, далее целую часть произведения $P_x \cdot 10000$ для файла Б и $P_y \cdot 10000$ для файла Б.

Решение

Для удобства восприятия разобьем наш код по частям:

1. Создадим функцию *dbscan*(), так как в задаче она нам понадобится несколько раз:

```
from math import dist
# Функция для кластеризации звёзд с использованием алгоритма DBSCAN
# a: Список звёзд, каждая из которых представлена как [x, y, масса]
# r: Радиус для определения соседства звёзд
# return: Список кластеров, каждый из которых представляет собой список звёзд
def dbscan(a, r):
    cl = []  # Список для хранения кластеров
    while a:  # Пока есть звёзды в списке
        cl.append([a.pop(0)])  # Начинаем новый кластер с первой звезды
        for j in cl[-1]:  # Для каждой звезды в текущем кластере
            for i in a:  # Проверяем каждую оставшуюся звезду
# Если расстояние между звёздами i и j меньше или равно r
                if dist(i, j) <= r:
                    cl[-1].append(j)  # Добавляем j в текущий кластер
                    a.remove(j)  # Удаляем j из списка оставшихся звёзд
    return cl  # Возвращаем список кластеров
```

2. Откроем наш файл, считаем его в список a, а затем воспользуемся созданной функцией *dbscan*(), чтобы разбить звезды на кластеры:

```
f = open('27.txt')
s = f.readline()
a = [list(map(float,i.replace(',','.').split())) for i in f]
r = 0.45
clusters = dbscan(a, r)
```

3. Теперь переходим к поиску тройной звездной системы.

Основная мысль заключается в том, что нам нужно пройти по каждой точке в четырех найденных кластерах и с помощью уже созданной функции *dbscan*() для каждого кластера найти списки звезд, расстояние между которыми менее 0.01.

Далее в каждом кластере нужно оставить только те списки, в которых количество звезд равно трем – то есть только тройные звездные системы. Причем отдельно нужно проверить, что в этой системе три звезды попарно находятся на расстоянии не более 0.01.

В конце остается дело за малым: нужно отобрать тройную звезду, которая образует треугольник с наименьшим периметром:

```
t = 0.01 # Расстояние для определения тройных звездных систем
sx = sy = 0 # Сумматоры для абсцисс и ординат
for i in clusters: # Проходимся по каждому кластеру
    if len(i) > 4: # Если в кластере >4 звезд, начинаем искать тройные звезды
        star_system = dbscan(i, t)
        min_p = 10000000000050000000
        for j in star_system:
            if len(j) == 3: # Проверяем, что система состоит из трех звезд
                t1 = dist(j[0], j[1]) # Попарно вычисляем расстояние
                t2 = dist(j[0], j[2])
                t3 = dist(j[1], j[2])
# Проверяем, что максимальное из попарных расстояний меньше 0.01
                if max(t1, t2, t3) <= t:
                    if t1 + t2 + t3 < min_p: # Ищем систему с мин. периметром
                        min_p = t1 + t2 + t3
                        good_system = j
# Считаем сумму абсцисс и ординат
        sx += good_system[0][0] + good_system[1][0] + good_system[2][0]
        sy += good_system[0][1] + good_system[1][1] + good_system[2][1]
print(int(sx / 12 * 10000), int(sy / 12 * 10000))
```

Совмещаем три части в полноценный код и получаем решение задачи:

```
from math import *
def dbscan(a,r):
    cl = []
    while a:
        cl.append([a.pop(0)])
        for j in cl[-1]:
            for i in a:
                if dist(i, j) <= r:
                    cl[-1].append(i)
                    a.remove(i)
    return cl
f = open('27.txt'), s = f.readline()
a = [list(map(float,i.replace(',','.').split())) for i in f]
r = 0.45, t = 0.01, sx = 0, sy = 0
clusters = dbscan(a, r)
for i in clusters:
    if len(i) > 4:
        star_system = dbscan(i, t)
        min_p = 10000000000050000000
        for j in star_system:
            if len(j) == 3:
                t1 = dist(j[0], j[1]), t2 = dist(j[0], j[2])
                t3 = dist(j[1], j[2])
                if max(t1, t2, t3) <= t:
                    if t1 + t2 + t3 < min_p:
                        min_p = t1 + t2 + t3
                        good_system = j
        sx += good_system[0][0] + good_system[1][0] + good_system[2][0]
        sy += good_system[0][1] + good_system[1][1] + good_system[2][1]
print(int(sx / 12 * 10000), int(sy / 12 * 10000))
```

Ответ: 58241 -3102