Яндекс. Тренировки по алгоритмам июнь 2021, занятие 5

J. Треугольники

	Все языки	Python 3.9.1
Ограничение времени	2 секунды	4 секунды
Ограничение памяти	256Mb	256Mb
Ввод	стандартный ввод или input.txt	
Вывод	стандартный вывод или output.txt	

Петя достаточно давно занимается в математическом кружке, поэтому он уже успел не только правила выполнения простейших операций, но и такое достаточно сложное понятие как симметрия. Для того, чтобы получше изучить симметрию Петя решил начать с наиболее простых геометрических фигур – треугольников. Он скоро понял, что осевой симметрией обладают так называемые равнобедренные треугольники. Поэтому теперь Петя ищет везде такие треугольники.
Напомним, что треугольник называется равнобедренным, если его площадь положительна, и у него есть хотя бы две равные стороны.

Недавно Петя, зайдя в класс, увидел, что на доске нарисовано n точек. Разумеется, он сразу задумался, сколько существует троек из этих точек, которые являются вершинами равнобедренных треугольников.

Требуется написать программу, решающую указанную задачу.

Формат ввода

Входной файл содержит целое число n (3 ≤ n ≤ 1500). Каждая из последующих строк содержит по два целых числа – x_i и y_i – координаты i-ой точки. Координаты точек не превосходят 10^9 по абсолютной величине. Среди заданных точек нет совпадающих.

Формат вывода

В выходной файл выведите ответ на задачу.

Пример 1

Ввод	Вывод
3 0 0 2 2 -2 2	1

Пример 2

Ввод	Вывод
4 0 0 1 1 1 0 0 1	4

Язык Python 3.9 (PyPy 7.3.11)

Набрать здесь | Отправить файл

```
    x, y = map(int, input().split())
    vertices.append((x, y))

answer = 0

# проходим по вершинам
for i in range(n):
    # просмотренные векторы
    used_vectors = set()
    # соседи текущей вершины
    neighbors = []
    # проходим по соседям вершины
    for j in range(n):
        # считаем координты вектора
        vec_x = vertices[i][0] - vertices[j][0]
        vec_y = vertices[i][1] - vertices[j][1]
        # считаем квадрат длины стороны
        square_side_len = vec_x**2 + vec_y**2
        # добавляем квадрат длины в массив neighbors
        neighbors.append(square_side_len)
        # если есть образующий прямую с текущим, то вычитаем 1 из answer
        if (-vec_x, -vec_y) in used_vectors:
            answer -= 1
        used_vectors.add((vec_x, vec_y))

    # сортируем по длине
    neighbors.sort()
    rg = 0
    # проходим левым указателем раздвижного окна по соседям
    for lf in range(len(neighbors)):
        # двигаем правый указатель до первого отличающегося соседа
        while rg < len(neighbors) and neighbors[lf] == neighbors[rg]:
            rg += 1
        # подсчитываем ответ
        answer += rg - lf - 1

# ответ
print(answer)
```

Отправить

Предыдущая