Физико-математический лицей № 239

**Поиск максимальной длины отрезка**

Отчет о домашней работе по информатике

Работу выполнила

Ученица 10-7 класса

Соловьева Катя

Санкт-Петербург

2021

**1. Постановка задачи**

На плоскости задано множество точек, и прямоугольник. Множество точек образует все возможные прямые, которые могут быть построены парами точек множества. Найти такую прямую (и такие две точки, через которые она проходит), что эта прямая пересекает указанный прямоугольник, и при этом длина отрезка прямой, находящейся внутри прямоугольника, максимальна.

В качестве ответа: выделить найденные две точки, нарисовать прямую, которая через них проходит, а также выделить на этой прямой отрезок между двумя найденными точками пересечения.

**2. Уточнение исходных и выходных данных и ограничений на них**

**2.1. Исходные данные**

Задан прямоугольник и множество точек.

Прямоугольник задается двумя вершинами одной из сторон. А также точкой, лежащей на прямой, проходящей через две другие вершины.

Точка задается своими координатами (x и y).

Прямоугольники и точки могут задаваться случайным образом, либо вводиться с клавиатуры.

Таким образом, получаем массив точек и один прямоугольник.

**2.2. Выходные данные**

Выделить найденные две точки, нарисовать прямую, которая через них проходит, а также выделить на этой прямой отрезок между двумя найденными точками пересечения.

**3. Выбор метода решения**

**3.1. Анализ исходных данных и выбор используемой структуры данных**

В задаче нужен массив точек,из которого будут выбираться случайные точки для задачи. Также должен быть доступен ввод точек с клавиатуры. Прямоугольник может задаваться либо тремя случайными точками из массива, либо вручную с клавиатуры.

**3.2. Выбор метода решения**

Получаем множество точек и прямоугольник. Находим длины отрезков прямых, образованных каждыми двумя точками множества, лежащих внутри прямоугольника. Находим максимальную длину такого отрезка. Выделяем две точки прямой, на которой лежит этот отрезок.

**4. Составление алгоритма**

1) Перебирая каждые две точки из множества точек, задаем прямые, проходящие через них.

2) Ищем, есть ли пересечение со сторонами прямоугольника.

3) Если есть, то по координатам точек ищем расстоянием между ними.

4) Выбираем наибольшее расстояние и смотрим, какие две точки задают эту прямую

5) Выделяем эти точки

**5. Листинг программы**

Метод, задающий случайный прямоугольник:

public static Rectangle getRandomRectangle() {

Random random = new Random();

double x = random.nextDouble() \* 2 - 1;

double y = random.nextDouble() \* 2 - 1;

double x1 = random.nextDouble() \* 2 - 1;

double y1 = random.nextDouble() \* 2 - 1;

double x2 = random.nextDouble() \* 2 - 1;

double y2 = random.nextDouble() \* 2 - 1;

return new Rectangle(new Vector2(x, y), new Vector2(x1, y1), new Vector2(x2, y2));

}

Основной метод, решающий задачу:

public void solve() {

Vector2 v1, v2;

// перебираем пары точек

double length = 0;

double maxlength = 0;

for (Point p : points) {

for (Point p2 : points) {

// если точки являются разными

if (p != p2) {

ArrayList<Vector2> intersectionPoints = rectangle.intersection2(

new Vector2(p.x, p.y), new Vector2(p2.x, p2.y)

);

if (intersectionPoints.size() != 2) length = 0;

} else {

Vector2 n = intersectionPoints.get(0);

Vector2 c = intersectionPoints.get(1);

length = Math.*sqrt*((n.x - c.x) \* (n.x - c.x) + (n.y - c.y) \* (n.y - c.y));

if (length >= maxlength) {

maxlength = length;

pointA = p;

pointB = p2;

}

}

}

}

}

}

Метод intersection2, возвращающий список точек пересечения прямой, образованной двумя случайными точками со сторонами прямоугольника, если такие существуют:

public ArrayList<Vector2> intersection2(Vector2 v, Vector2 v1) {

ArrayList<Vector2> lst = new ArrayList<>();

Line l = new Line(v.x, v.y, v1.x, v1.y);

Vector2 intersectionPoint = *isInQuad*(l, a, b);

if (intersectionPoint != null)

lst.add(intersectionPoint);

intersectionPoint = *isInQuad*(l, b, c);

if (intersectionPoint != null)

lst.add(intersectionPoint);

intersectionPoint = *isInQuad*(l, c, d);

if (intersectionPoint != null)

lst.add(intersectionPoint);

intersectionPoint = *isInQuad*(l, d, a);

if (intersectionPoint != null)

lst.add(intersectionPoint);

return lst;

}

Вспомогательный метод, проверяющий наличие точки пересечения прямой со сторонами прямоугольника:

public static Vector2 isInQuad(Line l, Vector2 a, Vector2 b) {

Line l1 = new Line(a.x, a.y, b.x, b.y);

Vector2 p = l1.intersection(l);

if (p == null)

return null;

Vector2 min = new Vector2(Math.*min*(a.x, b.x), Math.*min*(a.y, b.y));

Vector2 max = new Vector2(Math.*max*(a.x, b.x), Math.*max*(a.y, b.y));

if (p.x >= min.x && p.y >= min.y && p.x <= max.x && p.y <= max.y)

return p;

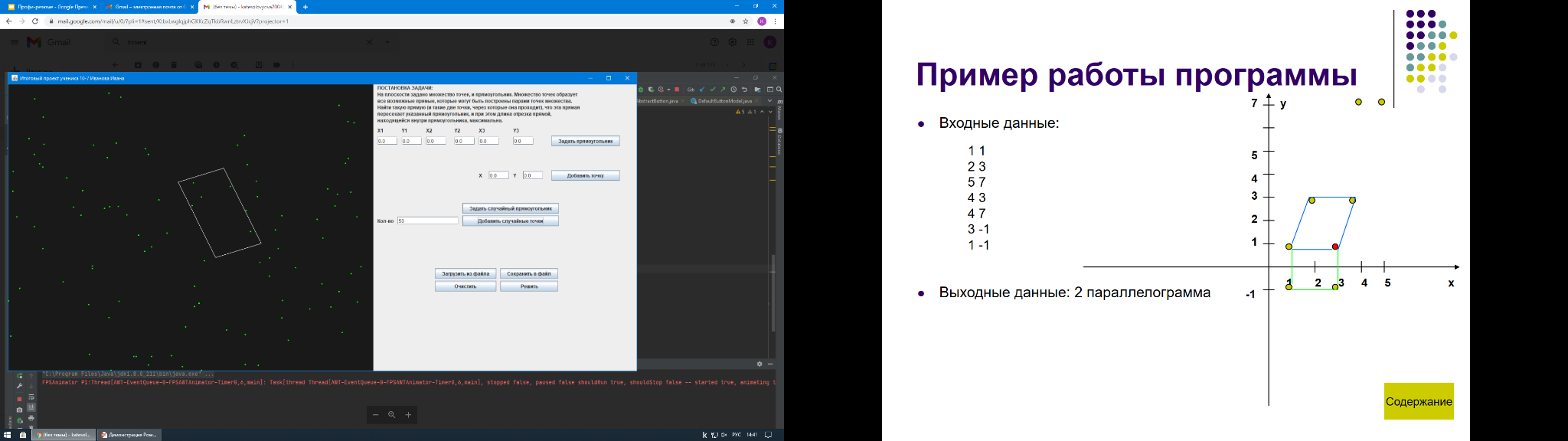
else

return null;

}

**6. Пример работы программы**

Получаем: массив из пятидесяти точек и прямоугольник, заданный тремя координатами.



На выходе получаем две искомые точки (выделены красным)

