**ГБОУ "Президентский ФМЛ № 239"**

**Нахождение прямой, проходящей через 2 точки из множества точек, с отрезком максимальной длины, лежащим внутри прямоугольника**

Годовой проект по информатике

Работу выполнил

Ученик 10-2 класса

Соловьяненко Егор

Санкт-Петербург

2021

**1. Постановка задачи**

На плоскости задано множество точек, и "параллельный" прямоугольник (прямоугольник, стороны которого параллельны осям координат). 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**

Исходные данные задаются двумя способами:

1) Чтение строк с данными из файла points&rectangle.txt. В первой строке задаются координаты противоположных вершин прямоугольника в виде 4 чисел, записанных через пробел. В остальных строках файла задаются координаты точек в виде 2 чисел на каждой строке, записанных через пробел, каждая пара координат отделяется переносом строки.

2) Ввод данных через интерфейс. Координаты точек задаются через 2 соответствующих поля, после чего надо нажать “Добавить точку”. Координаты прямоугольника вводятся через 4 соответствующих поля, после нужно нажать “Добавить прямоугольник”.

Координаты всех точек и прямоугольника ограничены промежутком [-1; 1] по осям абсцисс и ординат. Все координаты являются данными типа double с точностью до 2 знаков после запятой. Ввод координат осуществляется с отделением дробной части от целой с помощью запятой.

В случае, если введено менее 2 точек, то программа не выведет искомую прямую, т.к. для построения прямой необходимо 2 точки. Также необходимо ввести координаты прямоугольника, иначе он не будет задан. Следовательно, для верной работы программы нужно ввести хотя бы 2 точки и прямоугольник.

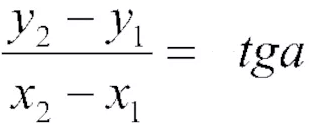
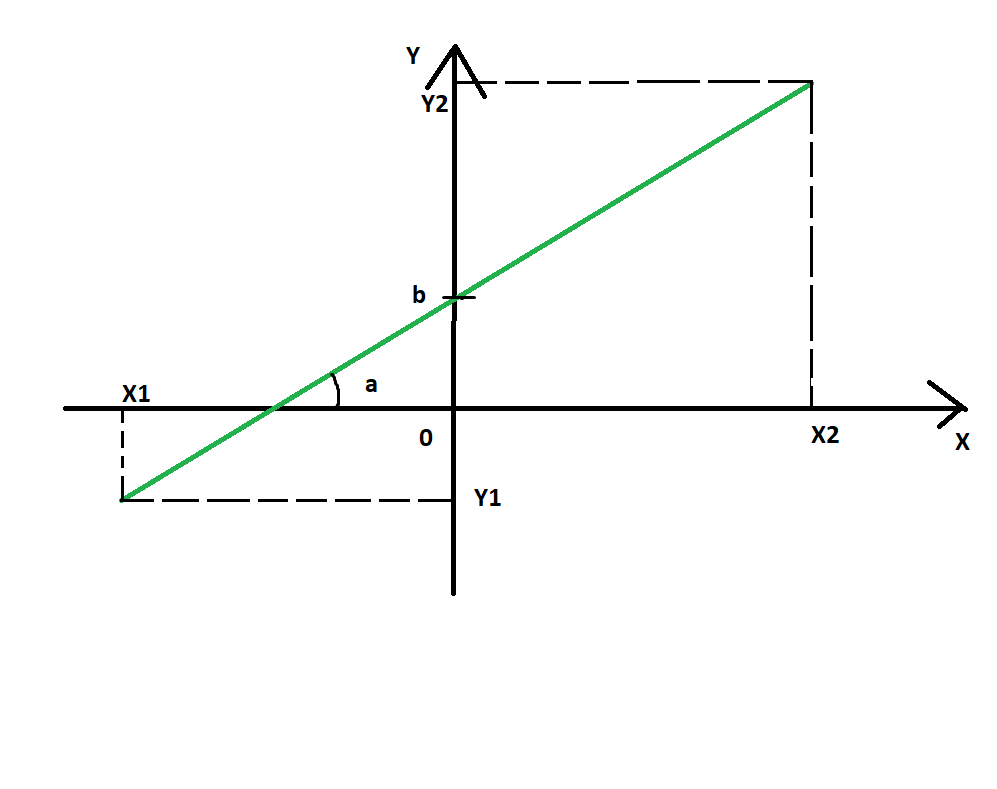
**2.2. Выходные данные**

Необходимо нарисовать искомую прямую на графической сцене, выделить на ней те 2 точки из заданного множества точек, через которые она проходит, и отрезок этой прямой, находящийся внутри прямоугольника.

**3. Математическая модель**

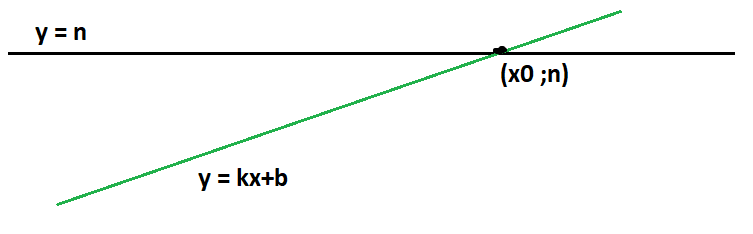
Для нахождения искомой прямой необходимо сначала вычислить угловой коэффициент рассматриваемой прямой и расстояние до оси абсцисс в точке пересечения оси ординат на основе уравнения прямой с угловым коэффициентом:. Учитывая, что k = tga, получаем, что

b = y1 – kx1

После этого надо найти точку пересечения этой прямой с прямыми, на которых лежат верхняя и нижняя стороны прямоугольника (прямые, параллельные оси абсцисс), а также его левая и правая стороны (прямые, параллельные оси ординат). Ордината, на которой находится верхняя/нижняя сторона прямоугольника известна (n = y1 для нижней стороны и n = y3 для верхней стороны), поэтому остается найти абсциссу точки пересечения:

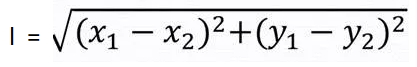


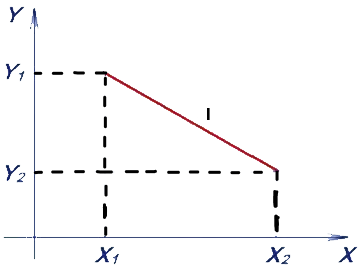


После найденная абсцисса точки пересечения сравнивается с границами стороны прямоугольника в процессе решения.

Для левой/правой стороны вычисления проводятся аналогично, только с поиском ординаты.

Когда мы нашли прямую, проходящую через прямоугольник, надо вычислить длину её отрезка, лежащей в прямоугольнике. Её найдем, зная точки пересечения.





**4. Анализ используемой структуры данных**

Точки задаются в динамическом массиве points объектами класса Point. Класс Point содержит координаты точки, заданные переменными x и y типа double, и параметр isSolution типа boolean, показывающий является ли точка искомой, определяющийся во время решения задачи.

Прямоугольник задается в динамическом массиве rectangles объектом класса Rectangle, являясь его единственным элементом. Класс Rectangle содержит координаты его первой и третьей вершин, заданные переменными x1, y1, x3, y3 типа double.

Искомая прямая и её отрезок внутри прямоугольника задаются в динамическом массиве lines объектами класса Line. Класс Line содержит координаты 2 точек, через которые проходит прямая, заданные переменными x1, y1, x2, y2 типа double, угловой коэффициент k типа double, вычисленный из значений x1, y1, x2 и y2, величину отрезка, отсекаемую прямой на оси ординат, считая от начала координат, b типа double, вычисленную из значений x1, y1 и k, и параметр inside типа boolean, показывающий является ли прямая искомым отрезком внутри прямоугольника, определяющийся во время решения задачи.

Для решения потребуются следующие переменные:

i, j - переменные циклов;

l, lmax - вычисленная и максимальная длины отрезка;

iold, jold - индексы точек последней найденной искомой прямой.

**5. Описание алгоритма**

1. Для решения требуется перебрать все пары точек, через которые пройдет прямая, с помощью вложенных циклов.
2. Определяются точки пересечения этой прямой сторон прямоугольника.
3. Если прямая пересекает прямоугольник в 2 точках, то вычисляется длина её отрезка, ограниченный точками пересечения прямой.
4. Вычисленная длина из пункта 3. сравнивается с уже найденной максимальной длиной, если вычисленная больше, максимальная к ней приравнивается.
5. Точки, через которые проводилась прямая, отмечаются как точки решения, а ранее отмеченные точки, наоборот, перестают отмечаться.
6. После перебора всех пар точек, на экран выводятся найденные прямая, её отрезок внутри прямоугольника и 2 точки, по которым строилась прямая.

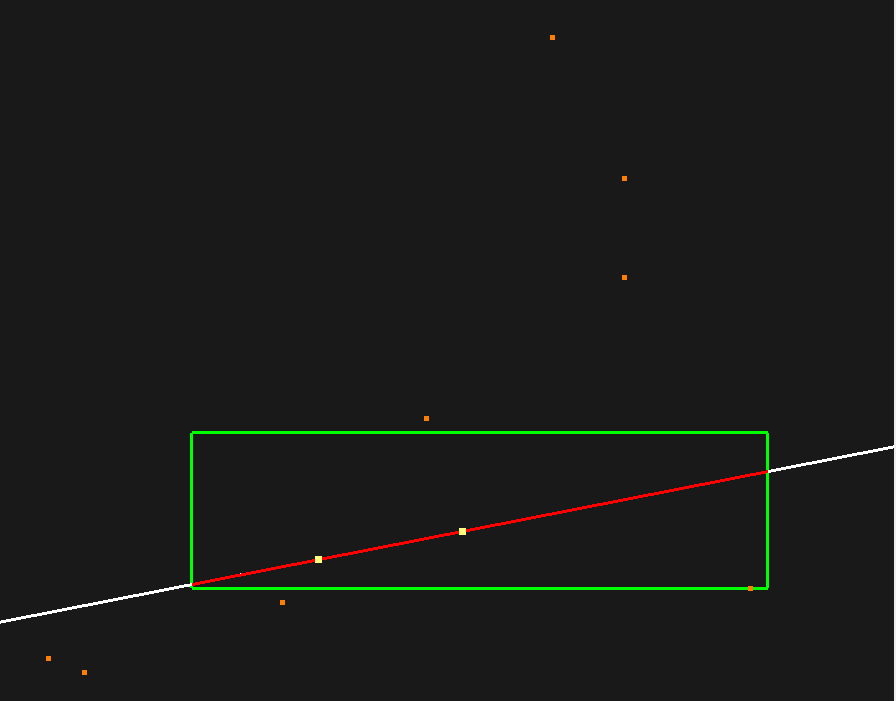
**6. Листинг программы**

public void solve() {  
 // обнуляем максимальную длину и индексы  
 double lmax = 0;  
 double l;  
 int iold = 0;  
 int jold = 0;  
 // каждую точку множества обозначаем как НЕ точку решения  
 for(Point point : points){  
 point.isSolution = false;  
 }  
 // если введено менее 2 точек  
 if(points.size() < 2) System.*out*.println("Введено недостаточно точек!");  
 // если не введен прямоугольник  
 if(rectangles.size() == 0) System.*out*.println("Прямоугольник не задан!");  
 else{  
 // перебираем пары точек  
 for (int i = 0; i < points.size(); i++){  
 Point p = points.get(i);  
 for (int j = 0; j < points.size(); j++){  
 Point p2 = points.get(j);  
 // если точки являются разными  
 if (p != p2) {  
 Rectangle rectangle = rectangles.get(0);  
 // создаем объекты прямой, проходящей через 2 точки, и её отрезок  
 Line line = new Line(p.x, p.y, p2.x, p2.y);  
 Line line2 = new Line(0,0,0,0);  
 // проверка на пересечение левой стороны  
 if(line.k \* rectangle.x1 + line.b <= Math.*max*(rectangle.y1, rectangle.y3) && line.k \* rectangle.x1 + line.b >= Math.*min*(rectangle.y1, rectangle.y3)){  
 line2.x1 = rectangle.x1;  
 line2.y1 = line.k \* rectangle.x1 + line.b;  
 }  
 // проверка на пересечение правой стороны  
 if(line.k \* rectangle.x3 + line.b <= Math.*max*(rectangle.y1, rectangle.y3) && line.k \* rectangle.x3 + line.b >= Math.*min*(rectangle.y1, rectangle.y3)){  
 // проверка, нашли ли уже первую точку пересечения  
 if(line2.x1 != 0 && line2.y1 != 0){  
 line2.x2 = rectangle.x3;  
 line2.y2 = line.k \* rectangle.x3 + line.b;  
 }  
 else{  
 line2.x1 = rectangle.x3;  
 line2.y1 = line.k \* rectangle.x3 + line.b;  
 }  
 }  
 // проверка, не нашли ли уже обе точки пересечения  
 if((line2.x1 == 0 && line2.y1 == 0) || (line2.x2 == 0 && line2.y2 == 0)){  
 // проверка на пересечение нижней стороны  
 if((rectangle.y1 - line.b) / line.k <= Math.*max*(rectangle.x1, rectangle.x3) && (rectangle.y1 - line.b) / line.k >= Math.*min*(rectangle.x1, rectangle.x3)){  
 // проверка, нашли ли уже первую точку пересечения  
 if(line2.x1 != 0 && line2.y1 != 0){  
 line2.x2 = (rectangle.y1 - line.b) / line.k;  
 line2.y2 = rectangle.y1;  
 }  
 else{  
 line2.x1 = (rectangle.y1 - line.b) / line.k;  
 line2.y1 = rectangle.y1;  
 }  
 }  
 }  
 // проверка, не нашли ли уже обе точки пересечения  
 if((line2.x1 == 0 && line2.y1 == 0) || (line2.x2 == 0 && line2.y2 == 0)){  
 // проверка на пересечение верхней стороны  
 if((rectangle.y3 - line.b) / line.k <= Math.*max*(rectangle.x1, rectangle.x3) && (rectangle.y3 - line.b) / line.k >= Math.*min*(rectangle.x1, rectangle.x3)){  
 // проверка, нашли ли уже первую точку пересечения  
 if(line2.x1 != 0 && line2.y1 != 0){  
 line2.x2 = (rectangle.y3 - line.b) / line.k;  
 line2.y2 = rectangle.y3;  
 }  
 else{  
 line2.x1 = (rectangle.y3 - line.b) / line.k;  
 line2.y1 = rectangle.y3;  
 }  
 }  
 }  
 // проверка, найдены ли две точки пересечения  
 if(line2.x1 != 0 && line2.y1 != 0 && line2.x2 != 0 && line2.y2 != 0){  
 // вычисление длины отрезка  
 l = Math.*sqrt*(Math.*pow*((line2.x1 - line2.x2), 2) + Math.*pow*((line2.y1 - line2.y2), 2));  
 if(l > lmax){  
 lmax = l;  
 // точки, через которые проходила последняя найденная прямая  
 points.get(iold).isSolution = false;  
 points.get(jold).isSolution = false;  
 // отмечаем пару точек как точки решения  
 iold = i;  
 jold = j;  
 p.isSolution = true;  
 p2.isSolution = true;  
 // отмечаем отрезок внутри прямоугольника как искомый  
 line.inside = false;  
 line2.inside = true;  
 // изменяем координаты прямой, чтобы она проходила по всей сцене  
 line.x1 = -1;  
 line.y1 = -line.k + line.b;  
 line.x2 = 1;  
 line.y2 = line.k + line.b;  
 // очищаем список прямых  
 lines.clear();  
 // добавляем в список искомые прямую и отрезок  
 lines.add(line);  
 lines.add(line2);  
 }  
 }  
 }  
 }  
 }  
 }  
 // если после перебора всех пар точек не нашлась искомая прямая  
 if(lines.size() == 0) System.*out*.println("При заданном множестве точек и прямоугольнике не существует искомой прямой!");  
}

**7.Пример работы программы**

**7.1 Входные данные**  **7.2 Выходные данные**

0,72 -0,24 -0,56 -0,68 0,04 -0,52  
-0,88 -0,88 -0,28 -0,60  
0,40 0,48 -0,56 -0,67  
0,04 -0,52 0,72 -0,35  
0,68 -0,68   
-0,36 -0,72   
-0,80 -0,92   
0,24 0,88   
0,40 0,20   
-0,28 -0,60   
-0,04 -0,20

**8. Анализ правильности решения**

Прямая, полученная в результате решения, всегда пересекает прямоугольник в 2 точках.

При добавлении точек к исходному их множеству отрезок пересекающей прямоугольник прямой, ограниченный его сторонами, стремится к диагонали прямоугольника, которая является отрезком максимальной длины, лежащим внутри прямоугольника.

Решение для 10 точек Решение для 60 точек

