Физико-математический лицей № 239

**Определение максимальной площади пересечения заданного множества «широких лучей»**

Отчёт о домашней работе по информатике

Работу выполнил

Ученик 10-7 класса

Лебедев Василий

Санкт-Петербург

2021

**1. Постановка задачи**

На плоскости задано множество «широких лучей». Найти такие два «широких луча», что фигура, находящаяся внутри этих двух «широких лучей» замкнута и имеет наибольшую площадь. В качестве ответа: выделить найденные два «широких луча», выделить контур фигуры, которая ограничивает точки внутри обоих «широких лучей», желательно выделить внутреннее пространство фигуры.

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

**2.1. Исходные данные**

Во входном файле beams.txt содержатся строки с координатами вершин «широких лучей в формате **x1 y1 x2 y2**. Например, **0.1 -1 0.4 -0.2.** Количество строк заранее неизвестно. Дробная часть может присутствовать, число может быть отрицательным, и это верно для всех вводимых чисел, так как это координаты. Проще говоря, ограничений на вводимые данные практически нет, кроме того, что там не должны присутствовать буквы и различные символы, не относящиеся к типу данных double.

**2.2. Выходные данные**

Необходимо вывести на экран введённое множество «широких лучей», выделить наибольшую область пересечения двух таких лучей и выделить сами эти лучи.

**3. Выбор метода решения**

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

Мы имеем координаты вершин каждого «широкого луча». Каждый из «широких лучей» нужно нарисовать, при этом «широкие лучи» могут добавляться, поэтому лучше всего создать класс WideBeams (для более удобного решения задачи) и создать список класса WideBeams, который будет расширяться по необходимости. У нас также есть максимальная область пересечения двух широких лучей, которая формально может иметь максимум 12 вершин (позже будет объяснено, почему именно 12). Поэтому создадим класс Polygon, объекты которого будут задаваться массивом точек (в нашем случае векторов) пересечения лучей. Вдобавок к этому, создадим объект этого класса, который мы будем рисовать.

**3.2. Выбор метода решения**

Считываем построчно координаты из файла, создаём из них объекты класса WideBeams и автоматически добавляем эти объекты в список. Создадим метод intersection в классе WideBeams, возвращающий массив векторов. В этот массив он будет записывать пересечения сторон двух «широких лучей» (которых максимум 8) и вершины этих «широких лучей», если они лежат внутри другого широкого луча, использующегося в этом методе (таких максимум 4). Поэтому массив имеет размер 12, все 12 векторов заполняются в том случае, если два «широких луча» совпадают. Всё это множество векторов образует необходимый нам многоугольник, являющийся площадью пересечения этих двух «широких лучей».

Однако найденное нами множество векторов может быть пронумеровано так, что многоугольник превратится в несколько других фигур, если концы этих векторов соединить по нумерации. Для того, чтобы решить эту проблему, создадим метод sort(), который перенумерует вектора в найденном множестве так, чтобы их концы образовывали площадь пересечения (выпуклый многоугольник).

Создадим метод Square() в классе Polygon, который будет высчитывать площадь пересечения при помощи формулы Гаусса.

Теперь остаётся перебрать все «широкие лучи» друг с другом при помощи двух циклов, найти наибольшую площадь пересечения, нарисовать её и выделить необходимые «широкие лучи».

**4.** **Составление алгоритма**

**4.1. Обобщённая блок-схема алгоритма**

Начало

Конец

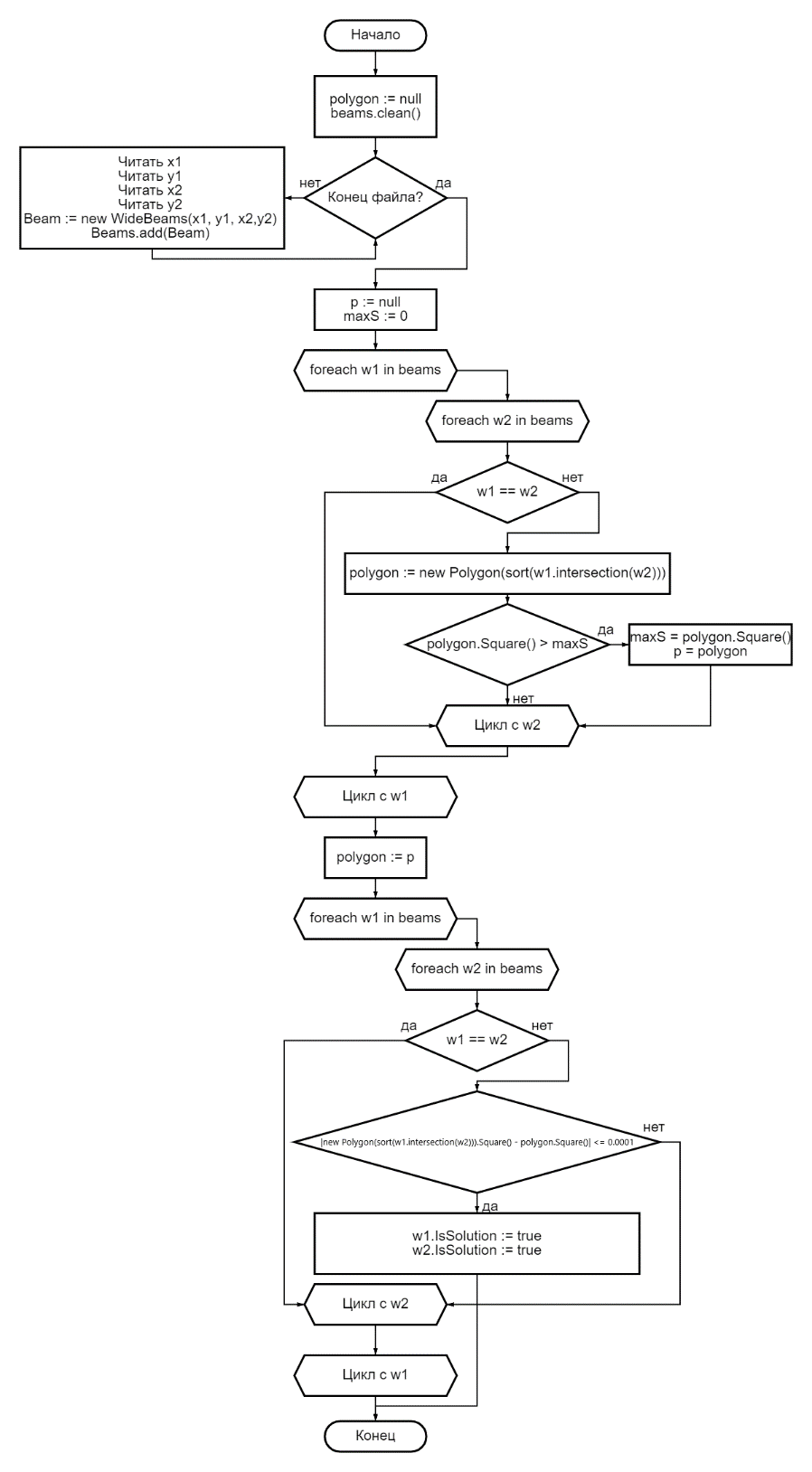
Закрашивание найденной площади. Выделение «широких лучей», образующих эту площадь.

Сравнение текущей площади пересечения с максимальной уже найденной площадью, фиксация максимальной и области ей соответствующей. К концу циклов перебора будет найдена максимальная площадь.

Нахождение каждого множества точек пересечения. Сортировка этого множества и создание из него области пересечения двух текущих «широких лучей».

После нажатия кнопки «Решить»: перебор всех «широких лучей».

Чтение исходных данных. Рисование множества «широких лучей».

** 4.2. Блок-схема алгоритма**

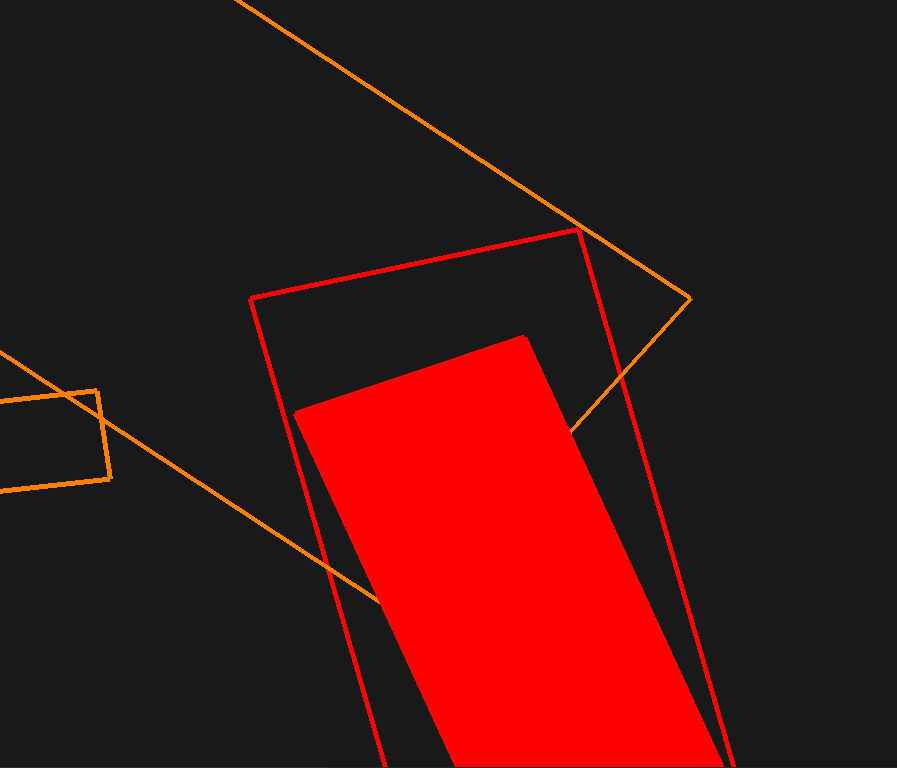
**5. Листинг программы**

*/\*\*  
 \* текст задачи  
 \*/*public static final String *PROBLEM\_TEXT* = "На плоскости задано множество \"широких лучей\". Найти такие два \"широких луча\",\n" +  
 "что фигура, находящаяся внутри этих двух \"широких лучей\" замкнута и имеет\n" +  
 "наибольшую площадь.\n" +  
 "В качестве ответа:\n" +  
 "выделить найденные два \"широких луча\",\n" +  
 "выделить контур фигуры, которая ограничивает точки внутри обоих \"широких\n" +  
 "лучей\",\n" +  
 "желательно выделить внутреннее пространство фигуры (залить цветом).";  
  
*/\*\*  
 \* заголовок окна  
 \*/*public static final String *PROBLEM\_CAPTION* = "Итоговый проект ученика 10-7 Василия Лебедева";  
  
*/\*\*  
 \* путь к файлу  
 \*/*private static final String *FILE\_NAME* = "beams.txt";  
  
*/\*\*  
 \* список широких лучей  
 \*/*private ArrayList<WideBeams> beams; //Список широких лучей  
  
private Polygon polygon; //Область максимальной площади пересечения  
  
*/\*\*  
 \* Решить задачу  
 \*/*public void solve() {  
 for (WideBeams w : beams){  
 w.IsSolution = false; //Перекрашиваем все лучи в стандартный цвет, если были добавлены новые, а старые не были убраны  
 }  
 double max = 0; //Площадь, которую мы будем фиксировать как максимальную  
 Polygon p = null; //Многоугольник, который мы будем фиксировать как максимальный  
 // Перебираем пары широких лучей из списка  
 for (WideBeams w1 : beams) {  
 for (WideBeams w2 : beams) {  
 if (w1 != w2) { // Если пара будет состоять их одного и того же широкого луча, то площадь будет считаться всей площадью луча, нам этого не надо  
 */\*\*  
 \* Метод w1.intersection(w2) ищет множество точек пересечения сторон двух "широких лучей" и точек одного луча, лежащих внутри другого  
 \* По сути, ищет точки(на самом деле вектора), образующие(концы которых образуют) область пересечения двух лучей  
 \* Метод sort(Vector[] v) сортирует найденные точки(концы векторов) так, чтобы они были пронумерованы по порядку вдоль периметра области пересечения  
 \* Второй метод сделан для удобства нахождения площади и отрисовки этой области  
 \*/* polygon = new Polygon(sort(w1.intersection(w2))); //Область пересечения тех лучей, которые мы перебираем  
 if (polygon.Square() > max){ //Если площадь найденной области больше максимальной уже найденной площади  
 max = polygon.Square(); //Обновляем значение максимальной площади  
 p = polygon; //Обновляем многоугольник, имеющий максимальную площадь  
 //Square() - метод нахождения площади многоугольника с отсортированными вершинами по формуле Гаусса  
 }  
 }  
 }  
 }  
 polygon = p; //Присваимаем отрисовываемому многоугольнику наибольшую область пересечения  
 for (WideBeams w1 : beams) { //Найдём решающие "широкие лучи", чтобы поменять их цвет  
 for (WideBeams w2 : beams) {  
 if (w1 != w2) {  
 //Если найденная площадь пересечения равна максимальной (с учётом погрешности), то это и есть решающие лучи  
 if (Math.*abs*(new Polygon(sort(w1.intersection(w2))).Square() - polygon.Square()) <= 0.0001){  
 w1.IsSolution = true;  
 w2.IsSolution = true;  
 // IsSolution - поле класса, отвечающее за цвет отрисовываемого "широкого луча"  
 break;  
 }  
 }  
 }  
 if (w1.IsSolution){ //Если решающие лучи найдены, то нет смысла продолжать их искать  
 break;  
 }  
 }  
}  
  
*/\*\*  
 \* Конструктор класса задачи  
 \*/*public Problem() {  
 beams = new ArrayList<>();  
}  
  
public void addBeam(double x, double y, double x2, double y2) {  
 WideBeams beam = new WideBeams(new Vector(x, y), new Vector(x2, y2));  
 beams.add(beam);  
}  
*/\*\*  
 \* Загрузить задачу из файла  
 \*/*public void loadFromFile() {  
 beams.clear();  
 try {  
 File file = new File(*FILE\_NAME*);  
 Scanner sc = new Scanner(file);  
 // пока в файле есть непрочитанные строки  
 while (sc.hasNextLine()) {  
 double x = sc.nextDouble();  
 double y = sc.nextDouble();  
 double x2 = sc.nextDouble();  
 double y2 = sc.nextDouble();  
 sc.nextLine();  
 WideBeams beam = new WideBeams(new Vector(x, y), new Vector(x2, y2));  
 beams.add(beam);  
 }  
 } catch (Exception ex) {  
 System.*out*.println("Ошибка чтения из файла: " + ex);  
 }  
}  
  
*/\*\*  
 \* Сохранить задачу в файл  
 \*/*public void saveToFile() {  
 try {  
 PrintWriter out = new PrintWriter(new FileWriter(*FILE\_NAME*));  
 for (WideBeams beam : beams) {  
 out.printf("%.2f %.2f %.2f %.2f \n", beam.a.x, beam.a.y, beam.b.x, beam.b.y);  
 }  
 out.close();  
 } catch (IOException ex) {  
 System.*out*.println("Ошибка записи в файл: " + ex);  
 }  
}

**6. Пример работы программы**

**6.1. Исходные данные**

0,29 0,40 -0,44 0,22   
-0,10 -0,61 0,54 0,22   
-0,75 -0,25 -0,78 -0,02   
0,17 0,12 -0,34 -0,08

**6.2 Выходные данные**