Физико-математический лицей № 239

Нахождение отрезка максимальной длины, высекаемого прямоугольников

Отчёт о годовом проекте по информатике

Автор:

Ученик 10-1 класса

Гладышев Никита

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.

2.Входные и выходные данные.

2.1.Входные данные.

В файле points.txt вводится координаты двух вершин прямоугольника и точки на противополохной стороне, точек на плоскости. Координаты лежат в промежутке от -1 до 1.

2.2.Выходные данные

На консоль выводятся коэффициенты, задающие искомую прямую, и координаты концов отрезка внутри прямоугольника.

3.Математическая модель.

Потребуется 2 формулы:

1. Формула координат точки пересечения двух прямых:

Где line – прямая, заданная точками, а l – прямая, содержащая сторону прямоугольника.

(line) Ax+By+C (l)Ax+By+C

. 🡨P

2.Формула длины отрезка:

Где x1,x2,y1,y2 – координаты точек пересечения.

(x1,y1)

(length) (x2,y2)

4.Анализ структуры данных.

Точки у нас хранятся в массиве ArrayPoints, а прямоугольник – в качестве переменной rect класса Rect. И то, и другое требуется хранить, так как их параметры участвуют в переборе каждых двух точек.

5.Выбор метода решения.

Для начала нужно проверить, пересекает ли прямая наш прямоугольник. Для этого смотрим, пересекает ли она прямые, содержащие стороны: проверим на «непараллельность», по формуле:выведем координаты точки пересечения и проверим, что её координата по OX находится между координатами каких-то двух вершин прямоугольника. Теперь, если предыдущие операции выполнились, мы можем найти расстояние между двумя точками пересечения прямой с прямоугольником и сравнить длину получившегося отрезка с максимальной. После этого, если длина нашего отрезка больше наибольшей на данный момент, сохраняем её в переменной max, а так же сохраняем концы отрезка данной длины. Таким образом, перебрав все пары точек, мы найдём отрезок с максимальной длиной и нужную прямую.

6.Листинг программы.

6.1.Класс отвечающий за точку.

public class Vector2 {

public double x;//задаём поле координаты по оси X

public double y;//задаём поле координаты по оси Y

public Vector2(double x,double y){

this.x=x;

this.y=y;

}

//задаём класс который опишет положение нашей точки на плоскости

public Vector2(Vector2 p){

this.x=p.x;

this.y=p.y;

}

//если нам известна точка, то не ищем её координаты, а сразу записываем в класс

}

6.2.Класс отвечающий за треугольник.

public class Triangle {

public Vector2 a;//поле для первой точки

public Vector2 b;//поле для второй тчоки

public Vector2 c;//поле для третьей точки

public Triangle(Vector2 a, Vector2 b, Vector2 c) {

this.a = a;

this.b = b;

this.c = c;

}

//задаём треугольник с вершинами в точках a,b,c

public Triangle(Point a, Point b, Point c) {

this.a = new Vector2(a.x, a.y);

this.b = new Vector2(b.x, b.y);

this.c = new Vector2(c.x, c.y);

}

//подставляем какие-то три точки из нашего списка,тем самым задавая его точное положеие в пространстве

public void render(GL2 gl) {

Figure.renderTriangle(gl, a, b, c, false);

}

//рисуем наш треугольник

public boolean regular() {

double r1 = Math.sqrt((a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y));//расстояние между первыми двумя точками

double r2 = Math.sqrt((c.x - b.x) \* (c.x - b.x) + (c.y - b.y) \* (c.y - b.y));//расстояние между вторыми двумя точками

double r3 = Math.sqrt((a.x - c.x) \* (a.x - c.x) + (a.y - c.y) \* (a.y - c.y));//расстояние между оставшимися двумя точками

return Math.abs(r1 - r2) < 0.01 && Math.abs(r1 - r3) < 0.01 && Math.abs(r2 - r3) < 0.01;//проверяем все 3 расстояния на равенство между собой

}

}

6.3.Класс рисования.

public class Figure {//класс, отвечающий за рисование

public static void renderPoint(GL2 gl,Vector2 pos,double size){//рисуем точку

gl.glPointSize((float) size);//задаём размер точки

gl.glBegin(GL2.GL\_POINTS);//оператор рисования точки

gl.glVertex2d(pos.x,pos.y);//задаём положение точкм

gl.glEnd();//конец рисования

}

public static void renderLine(GL2 gl,Vector2 posA,Vector2 posB,double width){//рисуем отрезок

gl.glLineWidth((float) width);//ширина линии отрезка

gl.glBegin(GL2.GL\_LINES);//оператор рисования отрезка

gl.glVertex2d(posA.x,posA.y);//начало отрезка

gl.glVertex2d(posB.x,posB.y);//конец отрезка

gl.glEnd();//конец рисования

}

public static void renderTriangle(GL2 gl,Vector2 posA,Vector2 posB,Vector2 posC,boolean filled) {//рисуем треугольник

if(!filled){//закрашенность треугольника

gl.glBegin(GL2.GL\_LINE\_STRIP);//оператор рисования линий по указанному порядку точек

gl.glVertex2d(posA.x,posA.y);//1 точка следования

gl.glVertex2d(posB.x,posB.y);//2 точка следования

gl.glVertex2d(posC.x,posC.y);//3 точка следования

gl.glVertex2d(posA.x,posA.y);//1 точка следования,так как нам нужно вернуться в начало,чтобы замкнуть треугольник

gl.glEnd();//конец рисования

}

}

}

6.4.Решение задачи

public void solve() {//решение задачи

triangles.clear();//очищаем картинку от треугольников в самом начале программы

boolean flg = true;

int a = 0;//вводим параметр,который будет считать треугольники для каждой точки

//в следующих строках для каждой точки считаем количество правильных треугольниках

for (int i = 0; i < points.size(); i++) {

Point p = points.get(i);//номер 1 точки

for (int j = 0; j < points.size(); j++) {

Point p2 = points.get(j);//номер второй точки

for (int k = 0; k < points.size(); k++) {

if (i == j || j == k || k == i)//проверка на то,что такой треугольник уже был

continue;//если был такой треугольник,то записываем его снова

Point p3 = points.get(k);//номер третьей точки

Triangle t = new Triangle(p, p2, p3);//создаём новый треугольник из этих трёх точек

if (t.regular()) {//проверка на правильность треугольника

triangles.add(t);//если он правильный,то добавляем его на картинку

a++;//увеличиваем число правильных треугольников для 1 точки

}

}

}//так как какие-то треугольники мы считали дважды, то мы будём не a, а a/2

if (a / 2 < 2) {//проверка на то,что условие дваждытреугольности выполняется

flg = false ;

}

a = 0;//чтобы рассматриваь другую точку заного,не учитывая предыдущих, нужно обнулить параметр

}

System.out.println(flg);//узнаём,является ли данное множество дваждытреугольным

}

7.Пример работы программы

Входные данные Выходные данные

0,35 0,59 true

-0,43 0,59

0,36 -0,76

-0,42 -0,76

-0,81 -0,09

-0,04 -0,08