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 прямоугольник.

Public class Rect {

|  |
| --- |
|  |
|  | double a1; |
|  | double a2; |
|  | double a3; |
|  | double a4; |
|  | double a5; |
|  | double a6; //координаты 3 точек: двух соседних вершин и точки на противоположной стороне |
|  | public Rect() { |
|  | this.a1=0; |
|  | this.a2=0; |
|  | this.a3=0; |
|  | this.a4=0.1; |
|  | this.a5=0.1; |
|  | this.a6=0.1; |
|  | } |
|  | public Rect(double a1, double a2, double a3, double a4, double a5, double a6){ |
|  | this.a1=a1; |
|  | this.a2=a2; |
|  | this.a3=a3; |
|  | this.a4=a4; |
|  | this.a5=a5; |
|  | this.a6=a6; |
|  | } // задаём прямоугольник по трём точкам |
|  | public static void renderRect(GL2 gl, double a1, double a2, double a3, double a4, double a5, double a6, int b){ |
|  | Line l = new Line(a1,a2,a3,a4); |
|  | Line l1 = new Line(a5,a6,a5 + l.A,a6 + l.B); |
|  | Line l2 = new Line(a5,a6,a5 + l1.A,a6 + l1.B); |
|  | Line lp1 = new Line(a1,a2,a1 + l.A,a2 + l.B); |
|  | Line lp2 = new Line(a3,a4,a3 + l.A,a4 + l.B); |
|  | Vector2 p = new Vector2((lp1.B\*l2.C-lp1.C\*l2.B)/(lp1.A\*l2.B-lp1.B\*l2.A), (lp1.A\*l2.C-lp1.C\*l2.A)/(lp1.B\*l2.A-lp1.A\*l2.B)); |
|  | Vector2 p1=new Vector2((lp2.B\*l2.C-lp2.C\*l2.B)/(lp2.A\*l2.B-lp2.B\*l2.A), (lp2.A\*l2.C-lp2.C\*l2.A)/(lp2.B\*l2.A-lp2.A\*l2.B)); |
|  | if(b==1) { |
|  | gl.glBegin(GL2.GL\_QUADS); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glVertex2d(a3, a4); |
|  | gl.glVertex2d(p1.x, p1.y); |
|  | gl.glVertex2d(p.x, p.y); |
|  | gl.glEnd(); |
|  | } else { |
|  | gl.glBegin(GL2.GL\_LINE\_STRIP); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glVertex2d(a3, a4); |
|  |  |
|  | gl.glVertex2d(a3, a4); |
|  | gl.glVertex2d(p1.x, p1.y); |
|  |  |
|  | gl.glVertex2d(p1.x, p1.y); |
|  | gl.glVertex2d(p.x, p.y); |
|  |  |
|  | gl.glVertex2d(p.x, p.y); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glEnd(); |
|  | } // рисуем прямоугольник, заданный по трём точкам |
|  | } |
|  | public void renderRect(GL2 gl){ |
|  | Line l = new Line(a1,a2,a3,a4); |
|  | Line l1 = new Line(a5,a6,a5 + l.A,a6 + l.B); |
|  | Line l2 = new Line(a5,a6,a5 + l1.A,a6 + l1.B); |
|  | Line lp1 = new Line(a1,a2,a1 + l.A,a2 + l.B); |
|  | Line lp2 = new Line(a3,a4,a3 + l.A,a4 + l.B); |
|  | Vector2 p = new Vector2((lp1.B\*l2.C-lp1.C\*l2.B)/(lp1.A\*l2.B-lp1.B\*l2.A), (lp1.A\*l2.C-lp1.C\*l2.A)/(lp1.B\*l2.A-lp1.A\*l2.B)); |
|  | Vector2 p1=new Vector2((lp2.B\*l2.C-lp2.C\*l2.B)/(lp2.A\*l2.B-lp2.B\*l2.A), (lp2.A\*l2.C-lp2.C\*l2.A)/(lp2.B\*l2.A-lp2.A\*l2.B)); |
|  | gl.glBegin(GL2.GL\_LINE\_STRIP); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glVertex2d(a3, a4); |
|  |  |
|  | gl.glVertex2d(a3, a4); |
|  | gl.glVertex2d(p1.x, p1.y); |
|  |  |
|  | gl.glVertex2d(p1.x, p1.y); |
|  | gl.glVertex2d(p.x, p.y); |
|  |  |
|  | gl.glVertex2d(p.x, p.y); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glEnd(); |
|  | } // рисуем текущий прямоугольник |
|  | public static void randomRect(GL2 gl, int b){ |
|  | Random r = new Random(); |
|  | double a1 = r.nextDouble()\*2-1; |
|  | double a2 = r.nextDouble()\*2-1; |
|  | double a3 = r.nextDouble()\*2-1; |
|  | double a4 = r.nextDouble()\*2-1; |
|  | double a5 = r.nextDouble()\*2-1; |
|  | double a6 = r.nextDouble()\*2-1; |
|  | Line l = new Line(a1,a2,a3,a4); |
|  | Line l1 = new Line(a5,a6,a5 + l.A,a6 + l.B); |
|  | Line l2 = new Line(a5,a6,a5 + l1.A,a6 + l1.B); |
|  | Line lp1 = new Line(a1,a2,a1 + l.A,a2 + l.B); |
|  | Line lp2 = new Line(a3,a4,a3 + l.A,a4 + l.B); |
|  | Vector2 p = new Vector2((lp1.B\*l2.C-lp1.C\*l2.B)/(lp1.A\*l2.B-lp1.B\*l2.A), (lp1.A\*l2.C-lp1.C\*l2.A)/(lp1.B\*l2.A-lp1.A\*l2.B)); |
|  | Vector2 p1=new Vector2((lp2.B\*l2.C-lp2.C\*l2.B)/(lp2.A\*l2.B-lp2.B\*l2.A), (lp2.A\*l2.C-lp2.C\*l2.A)/(lp2.B\*l2.A-lp2.A\*l2.B)); |
|  | if(b==1) { |
|  | gl.glBegin(GL2.GL\_QUADS); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glVertex2d(a3, a4); |
|  | gl.glVertex2d(p1.x, p1.y); |
|  | gl.glVertex2d(p.x, p.y); |
|  | gl.glEnd(); |
|  | } else { |
|  | gl.glBegin(GL2.GL\_LINE\_STRIP); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glVertex2d(a3, a4); |
|  |  |
|  | gl.glVertex2d(a3, a4); |
|  | gl.glVertex2d(p1.x, p1.y); |
|  |  |
|  | gl.glVertex2d(p1.x, p1.y); |
|  | gl.glVertex2d(p.x, p.y); |
|  |  |
|  | gl.glVertex2d(p.x, p.y); |
|  | gl.glVertex2d(a1, a2); |
|  | gl.glEnd(); |
|  | } // рисуем случайный прямоугольник |
|  | } |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  | } |

6.2.Класс отвечающий за прямую.

public class Line{

|  |
| --- |
|  |
|  |  |
|  | public double A; |
|  | public double B; |
|  | public double C; // коэффициенты прямой |
|  | public double x1; |
|  | public double y1; |
|  | public double x2; |
|  | public double y2; // координаты двух точек, задающих прямую |
|  |  |
|  |  |
|  | public Line() { |
|  | this.A = 0; |
|  | this.B = 0; |
|  | this.C = 0; |
|  | } |
|  | public Line(double x1, double y1, double x2, double y2) { |
|  | this.A = y1 - y2; |
|  | this.B = x2 - x1; |
|  | this.C = x1\*y2 - x2\*y1; |
|  | this.x1 = x1; |
|  | this.y1 = y1; |
|  | this.x2 = x2; |
|  | this.y2 = y2; |
|  | } // задать прямую по двум точкам |
|  | public void renderSect(GL2 gl, double size){ |
|  | gl.glPointSize((float) size); |
|  | gl.glBegin(GL2.GL\_LINES); |
|  | gl.glVertex2d(x1, y1); |
|  | gl.glVertex2d(x2, y2); |
|  | gl.glEnd(); |
|  | } // нарисовать отрезок по двум концам |
|  | public void renderLine(GL2 gl, double size){ |
|  | Vector2 v1 = new Vector2(x1,y1); |
|  | Vector2 v2 = new Vector2(x2,y2); |
|  | double w1 = x2-x1; |
|  | double w2 = y2-y1; |
|  |  |
|  | Vector2 v0 = new Vector2(w1/Math.sqrt(w1\*w1+w2\*w2), w2/Math.sqrt(w1\*w1+w2\*w2)); |
|  |  |
|  | double k1 = x2 + 3\*v0.x ; |
|  | double k2 = y2 + 3\*v0.y ; |
|  | double k3 = x2-3\*v0.x ; |
|  | double k4 = y2-3\*v0.y ; |
|  | gl.glPointSize((float) size); |
|  | gl.glBegin(GL2.GL\_LINES); |
|  | gl.glVertex2d(k1, k2); |
|  | gl.glVertex2d(k3, k4); |
|  | gl.glEnd(); |
|  | } // нарисовать прямую по двум точкам |
|  | } |

6.4.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

}

}

}

6.4.Решение задачи

public void solve{

|  |
| --- |
|  |
|  |
|  | double max = 0; // максимальная длина искомого отрезка |
|  | Point max\_p1 = new Point(5, 5); // точка пересечения прямоугольника с прямой |
|  | Point max\_p2 = new Point(5, 5); // точка пересечения прямоугольника с прямой |
|  |
|  |
|  | // перебираем пары точек |
|  | for (Point p : points) { |
|  | for (Point p2 : points) { |
|  | // если точки являются разными |
|  | if (p != p2) { |
|  | // если координаты у них не совпадают |
|  | if (Math.abs(p.x - p2.x) >= 0.01 && Math.abs(p.y - p2.y) > 0.01) { |
|  | Line l = new Line(rect.a1, rect.a2, rect.a3, rect.a4); |
|  | Line l1 = new Line(rect.a5, rect.a6, rect.a5 + l.A, rect.a6 + l.B); |
|  | Line l2 = new Line(rect.a5, rect.a6, rect.a5 + l1.A, rect.a6 + l1.B); |
|  | Line lp1 = new Line(rect.a1, rect.a2, rect.a1 + l.A, rect.a2 + l.B); |
|  | Line lp2 = new Line(rect.a3, rect.a4, rect.a3 + l.A, rect.a4 + l.B); |
|  | Point o2 = new Point((lp1.B \* l2.C - lp1.C \* l2.B) / (lp1.A \* l2.B - lp1.B \* l2.A), (lp1.A \* l2.C - lp1.C \* l2.A) / (lp1.B \* l2.A - lp1.A \* l2.B)); |
|  | Point o1 = new Point((lp2.B \* l2.C - lp2.C \* l2.B) / (lp2.A \* l2.B - lp2.B \* l2.A), (lp2.A \* l2.C - lp2.C \* l2.A) / (lp2.B \* l2.A - lp2.A \* l2.B)); |
|  | Line line = new Line(p.x, p.y, p2.x, p2.y); // задаём прямые, содержащие стороны прямоугольника, и точки – его вершины |
|  | Point inter1 = new Point(2, 2); |
|  | Point inter2 = new Point(2, 2); |
|  | Point inter3 = new Point(2, 2); |
|  | Point inter4 = new Point(2, 2); // задаём точки пересечения прямой и сторон прямоугольника |
|  | int par1 = 0; |
|  | int par2 = 0; |
|  | int par3 = 0; |
|  | int par4 = 0; // создаём параметры, с помощью которых поймём, пересекает ли прямая прямоугольник |
|  | boolean b; |
|  | if (line.B != l.B) { |
|  | b = Math.abs(l.A / l.B - line.A / line.B) < 0.0001; |
|  | } else { |
|  | b = 1 == 1; |
|  | } |
|  | if (b == (1 == 0)) { |
|  | inter1.y = (line.A \* l.C - line.C \* l.A) / (line.B \* l.A - line.A \* l.B); |
|  | inter1.x = (line.B \* l.C - line.C \* l.B) / (line.A \* l.B - line.B \* l.A); |
|  | } |
|  | if ((Math.abs(inter1.x - rect.a1) + Math.abs(inter1.x - rect.a3) == Math.abs(rect.a3 - rect.a1)) && (Math.abs(inter1.y - rect.a2) + Math.abs(inter1.y - rect.a4) == Math.abs(rect.a2 - rect.a4))) { |
|  | par1++; |
|  | } |
|  | if (line.B != lp2.B) { |
|  | b = Math.abs(lp2.A / lp2.B - line.A / line.B) < 0.0001; |
|  | } else { |
|  | b = 1 == 1; |
|  | } |
|  | if (b == (1 == 0)) { |
|  | inter2.y = (line.A \* lp2.C - line.C \* lp2.A) / (line.B \* lp2.A - line.A \* lp2.B); |
|  | inter2.x = (line.B \* lp2.C - line.C \* lp2.B) / (line.A \* lp2.B - line.B \* lp2.A); |
|  | } |
|  | if ((Math.abs(inter2.x - o1.x) + Math.abs(inter2.x - rect.a3) == Math.abs(rect.a3 - o1.x)) && (Math.abs(inter2.y - rect.a2) + Math.abs(inter2.y - o1.y) == Math.abs(rect.a2 - o1.y))) { |
|  | par2++; |
|  | } |
|  | if (line.B != l2.B) { |
|  | b = Math.abs(l2.A / l2.B - line.A / line.B) < 0.0001; |
|  | } else { |
|  | b = 1 == 1; |
|  | } |
|  | if (b == (1 == 0)) { |
|  | inter3.y = (line.A \* l2.C - line.C \* l2.A) / (line.B \* l2.A - line.A \* l2.B); |
|  | inter3.x = (line.B \* l2.C - line.C \* l2.B) / (line.A \* l2.B - line.B \* l2.A); |
|  | } |
|  | if ((Math.abs(inter3.x - o1.x) + Math.abs(inter3.x - o2.x) == Math.abs(o1.x - o2.x)) && (Math.abs(inter3.y - o1.y) + Math.abs(inter3.y - o2.y) == Math.abs(o1.y - o2.y))) { |
|  | par3++; |
|  | } |
|  | if (line.B != lp1.B) { |
|  | b = Math.abs(lp1.A / lp1.B - line.A / line.B) < 0.0001; |
|  | } else { |
|  | b = 1 == 1; |
|  | } |
|  | if (b == (1 == 0)) { |
|  | inter4.y = (line.A \* lp1.C - line.C \* lp1.A) / (line.B \* lp1.A - line.A \* lp1.B); |
|  | inter4.x = (line.B \* lp1.C - line.C \* lp1.B) / (line.A \* lp1.B - line.B \* lp1.A); |
|  | } |
|  | if ((Math.abs(inter4.x - rect.a1) + Math.abs(inter4.x - o2.x) == Math.abs(o2.x - rect.a1)) && (Math.abs(inter4.y - rect.a2) + Math.abs(inter4.y - o2.y) == Math.abs(rect.a2 - o2.y))) { |
|  | par4++; |
|  | } |
|  | if ((par1 != 0) && (par2 != 0)) { |
|  | if ((inter1.x - inter2.x) \* (inter1.x - inter2.x) + (inter1.y - inter2.y) \* (inter1.y - inter2.y) > max) { |
|  | max = (inter1.x - inter2.x) \* (inter1.x - inter2.x) + (inter1.y - inter2.y) \* (inter1.y - inter2.y); |
|  | max\_p1.x = inter1.x; |
|  | max\_p1.y = inter1.y; |
|  | max\_p2.x = inter2.x; |
|  | max\_p2.y = inter2.y; |
|  | } |
|  | } |
|  | if ((par1 != 0) && (par3 != 0)) { |
|  | if ((inter1.x - inter3.x) \* (inter1.x - inter3.x) + (inter1.y - inter3.y) \* (inter1.y - inter3.y) > max) { |
|  | max = (inter1.x - inter3.x) \* (inter1.x - inter3.x) + (inter1.y - inter3.y) \* (inter1.y - inter3.y); |
|  | max\_p1.x = inter1.x; |
|  | max\_p1.y = inter1.y; |
|  | max\_p2.x = inter3.x; |
|  | max\_p2.y = inter3.y; |
|  | } |
|  | } |
|  | if ((par1 != 0) && (par4 != 0)) { |
|  | if ((inter1.x - inter4.x) \* (inter1.x - inter4.x) + (inter1.y - inter4.y) \* (inter1.y - inter4.y) > max) { |
|  | max = (inter1.x - inter4.x) \* (inter1.x - inter4.x) + (inter1.y - inter4.y) \* (inter1.y - inter4.y); |
|  | max\_p1.x = inter1.x; |
|  | max\_p1.y = inter1.y; |
|  | max\_p2.x = inter4.x; |
|  | max\_p2.y = inter4.y; |
|  | } |
|  | } |
|  | if ((par2 != 0) && (par3 != 0)) { |
|  | if ((inter2.x - inter3.x) \* (inter2.x - inter3.x) + (inter2.y - inter3.y) \* (inter2.y - inter3.y) > max) { |
|  | max = (inter2.x - inter3.x) \* (inter2.x - inter3.x) + (inter2.y - inter3.y) \* (inter2.y - inter3.y); |
|  | max\_p1.x = inter2.x; |
|  | max\_p1.y = inter2.y; |
|  | max\_p2.x = inter3.x; |
|  | max\_p2.y = inter3.y; |
|  | } |
|  | } |
|  | if ((par2 != 0) && (par4 != 0)) { |
|  | if ((inter2.x - inter4.x) \* (inter2.x - inter4.x) + (inter2.y - inter4.y) \* (inter2.y - inter4.y) > max) { |
|  | max = (inter2.x - inter4.x) \* (inter2.x - inter4.x) + (inter2.y - inter4.y) \* (inter2.y - inter4.y); |
|  | max\_p1.x = inter2.x; |
|  | max\_p1.y = inter2.y; |
|  | max\_p2.x = inter4.x; |
|  | max\_p2.y = inter4.y; |
|  | } |
|  | } |
|  | if ((par3 != 0) && (par4 != 0)) { |
|  | if ((inter4.x - inter3.x) \* (inter4.x - inter3.x) + (inter4.y - inter3.y) \* (inter4.y - inter3.y) > max) { |
|  | max = (inter4.x - inter3.x) \* (inter4.x - inter3.x) + (inter4.y - inter3.y) \* (inter4.y - inter3.y); |
|  | max\_p1.x = inter4.x; |
|  | max\_p1.y = inter4.y; |
|  | max\_p2.x = inter3.x; |
|  | max\_p2.y = inter3.y; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } // проверяем, что точки пересечения есть, и находим их координаты, после чего ищем длину отрезка прямой, высекаемого прямоугольником.  // далее если он больше чем нынешний максимальный, то сохраняем его длину и точки – его концы |
|  | if (max\_p1.x == 5) { |
|  | System.out.println("ERROR"); // если пересечений нет или оно одно, выводим ошибку |
|  | } else { |
|  | System.out.println(max\_p1); |
|  | System.out.println(max\_p2); |
|  | resA = new Point(max\_p1.x, max\_p1.y); |
|  | resB = new Point(max\_p2.x, max\_p2.y); |
|  |  |
|  | resLine = new Line(resA.x, resA.y, resB.x, resB.y); |
|  | System.out.printf("resA.x = %.2f, resA.y = %.2f\n", resA.x, resA.y); |
|  | System.out.printf("resB.x = %.2f, resB.y = %.2f\n", resB.x, resB.y); |
|  | System.out.printf("resLine.A = %.2f,resLine.B = %.2f,resLine.C = %.2f\n", resLine.A, resLine.B, resLine.C); |
|  | } // если пересечения есть, выводим нужную прямую и точки-концы максимального отрезка. |
|  | } |
|  |  |

7.Пример работы программы

Ввод: Вывод:

-0,50 -0,63 -0,10 -0,54 0,81 0,27 resA.x=-0,23  
-0,84 -0,84 resA.y=0,03  
0,16 0,72 resB.x=-0,53

0,84 0,60 resB.y=-0,51  
0,08 0,76 resLine.A=0,54  
-0,60 -0,28 resLine.B=-0,3  
-0,80 -1,00 resLine.C=0,13  
-0,92 0,72  
0,80 0,84  
0,40 -0,44  
0,08 0,16