22.9 找到凸包的卷包裹算法

import javafx.geometry.Point2D;

import java.util.ArrayList;

public class Exercise21\_8 {

public static void main(String[] args) {

ArrayList<Point2D> list = new ArrayList<>();

list.add(new Point2D(2, 3));

list.add(new Point2D(8, 5));

list.add(new Point2D(0, 4));

list.add(new Point2D(2, 6));

list.add(new Point2D(7, 2));

list.add(new Point2D(9, 4));

list.add(new Point2D(5, 7));

ArrayList<Point2D> getIt = find(list);

for(Point2D temp : getIt)

System.out.println(temp.getX() + " " + temp.getY());

}

public static ArrayList<Point2D> find(ArrayList<Point2D> list) {

ArrayList<Point2D> wrapping = new ArrayList<>();

Point2D h0 = list.get(0);

Point2D t0;

Point2D t1;

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

Point2D temp = list.get(i);

if (temp.getX() \* temp.getX() + temp.getY() \* temp.getY() >

h0.getX() \* h0.getX() + h0.getY() \* h0.getY()) {

h0 = temp;

}

}

wrapping.add(h0);

t0 = h0;

list.remove(h0);

while(true){

if(list.size() == 0)

break;

t1 = list.get(0);

for (Point2D temp : list) {

if (judgeLorR(t0, t1, temp) > 0) {

t1 = temp;

}

}

if(t1 != h0) {

wrapping.add(t1);

list.remove(t1);

t0 = t1;

}

else

break;

}

return wrapping;

}

public static double judgeLorR(Point2D s1, Point2D s2, Point2D s0){

Point2D p1 = new Point2D(s1.getX() - s2.getX(), s1.getY() - s2.getY());

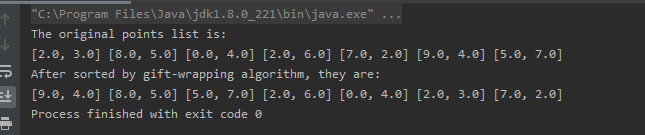
Point2D p2 = new Point2D(s0.getX() - s1.getX(), s0.getY() - s1.getY());

return p1.getX() \* p2.getY() - p2.getX() \* p1.getY();

}

}

运行结果：



22.11 寻找凸包的格雷厄姆算法

import java.util.ArrayList;

import java.util.Stack;

public class Exercise21\_11 {

public static void main(String[] args){

double[][] list = {{2, 3}, {8, 5}, {0, 4}, {2, 6}, {7, 2}, {9, 4}, {5, 7}};

System.out.println("The original points list is: ");

for(double[] temp : list)

System.out.print("[" + temp[0] + ", " + temp[1] + "] , ");

System.out.println();

System.out.println("After sorted by Graham algorithm, they are: ");

Stack<MyPoint> getIt = getConvexHull(list);

for(MyPoint temp : getIt)

System.out.println("[" + temp.x + ", " + temp.y + "]");

}

private static class MyPoint implements Comparable<MyPoint>{

double x, y;

MyPoint(double x, double y){

this.x = x;

this.y = y;

}

@Override

public int compareTo(MyPoint o) {

if((this.x \* this.x + this.y \* this.y) > (o.x \* o.x +o.y + o.y))

return 1;

else if((this.x \* this.x + this.y \* this.y) == (o.x \* o.x +o.y + o.y))

return 0;

else

return -1;

}

}

public static Stack<MyPoint> getConvexHull(double[][] s){

ArrayList<MyPoint> points = new ArrayList<>();

ArrayList<MyPoint> getIt = new ArrayList<>();

for(double[] temp : s){ //Insert the ArrayList

points.add(new MyPoint(temp[0], temp[1]));

}

MyPoint rightDown = points.get(0);

for(MyPoint temp : points){ //Find the rightDown point

if(temp.compareTo(rightDown) > 0)

rightDown = temp;

}

sort(points); //Sort the points ArrayList

Stack<MyPoint> H = new Stack<>();

H.push(rightDown);

H.push(points.get(0));

H.push(points.get(1));

H.push(points.get(2));

for(int i = 3; i < points.size()-1; ) {

MyPoint t1 = H.pop();

MyPoint t2 = H.peek();

H.push(t1);

if (judgeLorR(t1, t2, points.get(i)) < 0) {

H.push(points.get(i));

i++;

} else {

getIt.add(H.pop());

}

}

return H;

}

public static void sort(ArrayList<MyPoint> list){

MyPoint temp;

MyPoint p0 = list.get(0);

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

for(int j = 1; j < list.size() - i; j++){

if(judgeBorS(list.get(j-1), list.get(j), p0) > 0){

temp = list.get(j-1);

list.set(j-1, list.get(j));

list.set(j, temp);

}

else if(judgeBorS(list.get(j-1), list.get(j), p0) == 0){

double distance1 = Math.sqrt(Math.pow(list.get(j).x - p0.x, 2) +

Math.pow(list.get(j).y - p0.y, 2));

double distance2 = Math.sqrt(Math.pow(list.get(j-1).x - p0.x, 2) +

Math.pow(list.get(j-1).y - p0.y, 2));

if(distance1 > distance2)

list.remove(j-1);

else

list.remove(j);

}

}

}

}

public static int judgeBorS(MyPoint s1, MyPoint s2, MyPoint s0){

double angle1 = (s1.x - s0.x) / (Math.sqrt(Math.pow(s1.x - s0.x, 2) +

Math.pow(s1.y - s0.y, 2)));

double angle2 = (s2.x - s0.x) / (Math.sqrt(Math.pow(s2.x - s0.x, 2) +

Math.pow(s2.y - s0.y, 2)));

if(s2.y - s0.y < 0 && s1.y - s0.y > 0)

return 1;

else if (s2.y - s0.y > 0 && s1.y - s0.y < 0)

return -1;

else if(s2.y - s0.y >= 0 && s1.y - s0.y >= 0) {

if (angle1 > angle2)

return 1;

else if (angle1 == angle2)

return 0;

else

return -1;

}

else{

if (angle1 > angle2)

return -1;

else if (angle1 == angle2)

return 0;

else

return 1;

}

}

public static double judgeLorR(MyPoint s1, MyPoint s2, MyPoint s0){

MyPoint p1 = new MyPoint(0, 0);

MyPoint p2 = new MyPoint(0, 0);

p1.x = s1.x - s2.x;

p1.y = s1.y - s2.y;

p2.x = s0.x - s1.x;

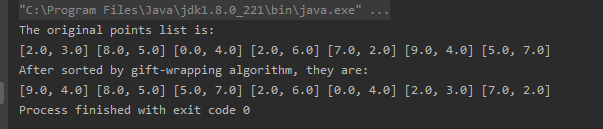
p2.y = s0.y - s1.y;

return p1.x \* p2.y - p2.x \* p1.y;

}

}

运行结果：



22.12 最后的100个素数

import java.io.BufferedInputStream;

import java.io.DataInputStream;

import java.io.FileInputStream;

public class Exercise22\_12 {

public static void main(String[] args) throws Exception {

DataInputStream input =

new DataInputStream(new BufferedInputStream(

new FileInputStream("PrimeNumbers.dat")));

input.skip(input.available() - 8 \* 100);

int inputEnter = 0;

while (input.available() > 0) {

System.out.print(input.readLong() + " ");

inputEnter++;

if(inputEnter % 10 == 0)

System.out.println();

}

input.close();

}

}

运行结果：

