CS 430 Programming Languages
Spring 2017
Module 7 PA

JAMES MADISON UNIVERSITY

Introduction

In this PA you will write several classes for storing and manipulating 2D points, pairs of points (which are also edges of polygons), and sets of points. These classes will host methods that compute the closest pair of points from a set of points, and the convex hull of a set of points. A *convex polygon* is closed figure such that a line drawn between any two points in or on the polygon does not include any points outside the polygon. The *convex hull* of a set of points is the smallest convex polygon that includes the set of points. If you imagine a nail driven into the plane at each point in the point set, and then imagine a rubber band placed around the nails so that every nail is inside the rubber band, the rubber band forms the convex hull.

Requirements

Your program will have three classes.

The `Point` class will have two instance attributes `x` and `y` (that is, `x` and `y` will be `attr_accessor` instance variables). `Point` must have the following methods.

`initialize(x, y)`—must set `@x` and `@y`.

`==(p)`—must redefine this equality operator so that it indicates whether the `self` and another `Point` instance have the same `@x` and `@y` values.

`distance(p)`—must compute the Euclidean distance between the `self` and another `Point`.

`sqr_distance(p)`—must compute the square of the Euclidean distance between the `self` and another `Point`. This is more efficient than computing the actual distance and can be used instead of the actual distance to find closest pairs of points.

`turn_direction(p1, p2)`—must return 1 if the `self Point` is to the right when moving from `Point p1` to `Point p2`, `-1` if it is to the left, and 0 if it is on the line formed by `p1` and `p2`. The body of this method is the following:

```
0 <=> (p2.x-p1.x)*(@y-p1.y) - (@x-p1.x)*(p2.y-p1.y)
```

This method is used in finding convex hull edges.

The `Pair` class will have two instance attributes `p1` and `p2`, each of which is a `Point`. `Pair` must have the following methods.

`initialize(p1, p2)`—must set `@p1` and `@p2`.

`==(pair)`—must redefine this operator so that two pairs are considered the same if they have the same points, in either order.

The `Pair` class has the alias `Edge`, which you can establish by simply defining the constant `Edge` to be `Pair`.

The Point_Set class will have an instance variable holding a list of points that will be treated like a set. This list must not be an attribute of any kind. It must have the following methods.

initialize(points = [])—must set the private list of points to points, or [] by default.

size—must return the number of points in the set.

add(p)—must add a Point p to the set, but do nothing if p is already in the set.

include?(p)—must return true just in case Point p is in the set.

all_pairs—must return a list of all the Pairs formed from the Points in the set, with no duplicates. If there are less than two Points in the set, then it must return [].

closest_pair—must return a closest pair of Points in the set. If the set has fewer than two Points, it must return nil. If the set has more than one closest pair, any one of them can be returned. This method must use Point.sqr_distance to determine a closest pair.

is_hull_edge(edge)—must return true just in case the edge (which is an instance of Pair/Edge) is part of the convex hull of the set of Points. Note that an edge is part of the convex hull just in case the turn direction of all the points in the set from that edge is either >= 0 or <= 0.

convex_hull—must return a list of all the Edges in the convex hull of the Point_Set, with no repetitions (that is, each Edge must occur exactly once). If the Point_Set has fewer than two points, this method must return [].

Your class must not contain any Ruby testing framework test classes. I will be using my own test classes to test your code and I don't want to have to mess with yours.

Advice

You are advised to write test cases for your code as you develop it. Your test code can be in another file or in the file you turn in, but commented out before you submit it.

You might find it useful to make to_s methods for your classes (this is the Ruby equivalent of the toString() method in Java). Then you can print out your classes in a more readable form when debugging.

Deliverable Requirements

You must name your ruby script file "mod7PA.rb". All the code must appear in this file. You must put your name in a comment at the top of the file. As usual, decompose your methods to make them more readable, use good names, indent properly, and so on. You will partly be graded on the readability of your code but mostly on whether it works.

Deliver your code by the deadline on Canvas. You will only be able to upload files with the extension .rb.