Object-Oriented Programming
Homework 2 Documentation

Iova Daniel-Alexandru
CEN 2.2B

Table of contents

Problem statement

4. Design and implement a module for polygons. The module must allow to solve the following problems related to polygons:
- Creating of a polygon;
- Printing of a polygon;
- Determining the rectangle with a minimum area, which contains inside the polygon;
- Determining the area of a polygon;
- Determining the perimeter of a polygon;
- Determining the gravity center;
- Determining if the polygon is convex;
- Comparison of polygons according to their area;
- Concatenation of two polygons; the concatenation of two polygons is a convex polygon obtained as following:
- Determines the reunion of the two sets containing the vertices of the two polygons;
- Determines the convex-hull of the set of vertices above determined;
- The vertices of the convex-hull represent the vertices of the new polygon;

Important Things to Note

The functions inside the polygon() class have the exact same functionality as in the polygon module from homework 1, but were adjusted to use the members of the polygon() class.

The testing functions are the exact same as in homework 1, the only modification being that they now take and return polygon() type objects. Thus, the algorithms and testing functions will not be explained again.

Polygon Class

The polygon class has two members:

1. Point List
2. Point Number

Constructor

Here we assign a list to the point list, then assign it's length to the point number.

Creating a polygon

Polygon creation is done in the following steps:

1. Read the number of points the polygon will have (we'll call it n) and assign it to the object's point number
2. Read the coordinates of the n points. The format of the input will be "{x} {y}"
3. Append a list of [x, y] format to the object's point list

Printing a polygon

Polygon printing is done through the following steps:

1. Iterate through the points of the object's point list
2. For each point, output a line of format "Point[{index}]'s coordinates: {x} {y}", where:
    a. index is the index of the current point
    b. x is the first element of the point
    c. y is the second element of the point

Determining the bounding box

The bounding box is computed as follows:

1. Compute the bottom left point:
    a. The x coordinate will be the minimum of all x coordinates (we'll call it left_x)
    b. The y coordinate will be the minimum of all y coordinates (we'll call it low_y)
2. Compute the top right point:
    a. The x coordinate will be the maximum of all x coordinates (we'll call it right_x)
    b. The y coordinate will be the maximum of all y coordinates (we'll call it high_y)
3. The bounding box will have the following coordinates (or any rotation of these points):
    a. [left_x, high_y]

   b. [right_x, high_y]
   c. [right_x, low_y]
   d. [left_x, low_y]
4. We return the bounding box, which is a polygon() object with the above point list (so a list of four lists, each containing two elements).

The minimum and maximum are done using python's max, min and list comprehension tools.

Determining the area of a polygon

The area was calculated using the shoelace method.
The function iterates through the current object's point list to compute the formula below:

$$\text{Area} = \frac{1}{2}\left|\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_{i1})\right|$$

The returned area will be of float type.

Determining the perimeter of a polygon

Determining the perimeter was done using 2 functions:

1. The first one determines the distance between 2 points
2. The second uses the first to calculate the perimeter

The first function:

1. It receives two points as arguments (other than the implicit self parameter)
2. Returns the distance between them using the formula $\sqrt{(x_B - x_A)^2 + (Y_B - Y_A)^2}$

The second function iterates through the current object's point list and uses the above formula to compute the perimeter.

The returned perimeter will be of float type.

Determining the gravity center of a polygon

Determining the gravity center uses the area of the polygon in the following formulas:

$$C_X = \frac{1}{6 \cdot Area} \left| \sum_{i=0}^{n-1} (x_i + x_{i+1}) \cdot (x_i y_{i+1} - x_{i+1} y_{i1}) \right|$$

$$C_Y = \frac{1}{6 \cdot Area} \left| \sum_{i=0}^{n-1} (y_i + y_{i+1}) \cdot (x_i y_{i+1} - x_{i+1} y_{i1}) \right|$$

The function iterates through the current object's point list and uses the above formulas to compute the coordinates of the gravity center.

The gravity center computation will not work with self-intersecting polygons.

Determining if a polygon is convex

Determining the convexity was done using 2 functions:

1. The first one determines the orientation of three consecutive points
2. The second uses the first to compute the polygon convexity

The orientation function was explained in the first homework's documentation.

The convexity function takes a polygon() object uses the orientation function to determine if the current polygon is convex or not (if yes returns true, if no returns false).

Comparison of polygons according to their area

Simple function which takes two polygon() objects.
Returns if the area of the first polygon is greater than the area of the second.

Concatenation of two polygons

Concatenating two polygons is done as follows:

1. Make a reunion of the two input polygons
2. Return the convex hull of the reunion

Making a reunion is straight forward, we create a new polygon() object to which we pass the current object's point list united with the second object's point list.

For the convex hull, we use the Jarvis' March Algorithm and return a new polygon() object with the computed hull list as a parameter to it's constructor. The function was explained in the first homework's documentation.

Menu function

Everything is connected by the menu function present in the main file. This receives the polygon object, a polygon() type object with the point list starting as a.The menu works like an infinite switch statement, with the initial option being -1 and then running the menu until the user presses 0. Inside the menu, the options are displayed and user input is requested. For each option their respective test function is run.