Artificial Intelligence Homework 1

Levi Crider
2/9/2023
lccrider@uark.edu
010914822

Abstract

The overall goal of this assignment was to solidify our foundational python and programming skills. I did this by implementing classes, using the four pillars of object-oriented programming, implementing formulas, and lastly by implementing algorithms.

Geometry Implementations

Geometry Class

To begin the project, I implemented a geometry class and then five subclasses, each unique shape: Triangle, Rectangle, Square, Circle, and Polygon. I had to implement a constructor, use the super constructor, and a calculate_area() method in each subclass. To start, I implemented the Geometry class itself. The implementation involved creating a constructor that would set the default shape name to "Shape" and take in points passed in for each shape. Afterward, it would set these variables as class variables. In the geometry class, I also had a method named get_name() which allowed us to return the name of the current shape object being called. count_number_of_geometry() was another class method. This method will enable us to see how many Geometry objects have been instantiated. Lastly, after seeing the review lecture, I also implemented a distance() method. This method would return the distance between two points, which proved useful later.

Triangle Class

After creating the base geometry class, the task was to create our first shape class, Triangle. I want to inherit the methods and constructor I used in our Geometry class, so I passed Geometry as a parameter to our Triangle class since inheritance works in python. Next, it was necessary to create our constructor and call the super constructor. The constructor consisted of parameters a, b, and c. Each of these points was a tuple representing a vertex of the Triangle. I passed these tuples into my super constructor to store the points in self.points, which allowed me to use them in the calculate_area() function that was implemented. The calculate area function was relatively easy to implement since all that was necessary was to get the length of the three sides using our distance formula and then use Heron's formula, $A = \sqrt{s(s-a)(s-b)(s-c)}$.

Rectangle Class

The rectangle class was also reasonably easy to implement. Creating the constructor was the same process as in the triangle class, but the calculate_area() method was different. The rectangle area formula is simple, $length * height$, so to implement this, I calculated the length from the points given and the height from the points given and returned the above formula as the area.

Square Class

The square class implementation is simple. Instead of passing Geometry as the parent class, the Rectangle class will now be the parent class. This made the square class implementation very simple. I could have used the calculate_area() method from the rectangle class, but instead, I created a unique method for the Square class. I was given points and side lengths for the square, but each side length is the same in a square. So to implement the calculate_area() method, I returned $length^2$.

Circle Class

Circle is another simple implementation. This time the constructor takes in Geometry as the parent class again. I am given a point and radius for the Circle as parameters which I pass into the super constructor in my Circle's class constructor. After this, all that needed to be done was the calculate_area() method. To implement this, I knew that the area of a circle was $\pi \times r^2$. Since radius was one of the variables passed into the constructor, all I had to do was use NumPy.pi. I then returned the formula above.

Poloygon Class

The last Geometry class to implement was the Polygon class. This class was a bit trickier, but it was easy after looking up the area formula. The constructor implementation was the same as the circle constructor, except I was given each point on the polygon. The last step was to create the calculate_area() function. The overall goal was to break up our polygon into triangles and then use Heron's formula (listed above in the Triangle class) once again to calculate the total area of our polygon by looping from 2 to N, where N is the number of sides of the polygon.

Matrix Multiplication Implementation

The next task was to implement a function to do matrix multiplication without using NumPy's matrix multiplication functionality. This was achieved by first getting the shape of the two matrices. To do this, I used the .shape() method on both of the matrices passed into the function. I set the rows and columns of each matrix equal to rowsA, rowsB, colsA, and colsB, respectively. After this, it was necessary to create a matrix named C, consisting of all 0's that had the same number of rows as matrix A and columns as matrix B due to how the final product of matrix multiplication comes out. The last step in this function was multiplying matrix A's row elements by each matrix B's column elements. This was possible by implementing a double for-loop.

Powers Of A Matrix Implementation

The following function to implement was pow(), which takes in a matrix and an integer n. This function will return the passed in matrix A to the power of n as shown here: A^n. The goal, however, was to implement this function once recursively and once iteratively.

To implement the recursive $pow(A, n)$ function, it was necessary to start with our base case. If n were equal to 0, I would return a matrix of all 1's, as anything to the power of 0 is 1. After this, I used a divide-and-conquer recursive algorithm instead of multiplying our matrix together n times. To do this, a matrix B was set to equal $pow(A, int(n/2))$. After this recursive call, it is essential to check whether our n is even or odd. If it is even, it is only necessary to return matrix B. If it is odd, however, it is necessary to return B multiplied by A.

The iterative $pow(A, n)$ function was slightly different but involved some of the same steps. I still created a matrix named result filled with 0's with the same number of rows and columns as matrix A that is passed into the function. After this, it was necessary to check if the power was odd, and if it was, $result = result * A$. Next, $A = A * A$ and divide n by 2 to decrease the number of times we have to do matrix multiplication. Once this is done looping, all that is left to do is return $result$.

Fibonacci Implementation

The Fibonacci implementations were quite simple. We had already created the functions we needed to implement these methods. The formulas were also given to us. So all that was necessary was to implement the formula using the python programming language. The results of these two functions will be shown in the results section below.

Implementations For Searching Algorithms

The final few functions done on this assignment were the depth-first search, recursive searching algorithms, breadth-first search, an iterative queue-based searching algorithm, and a find minimum, which required a priority queue. Depth and breadth were search algorithms I implemented in programming foundations two, but it had been quite a while. After researching and attending the lectures, I knew the concepts needed to be implemented. In depth-first search, we want to search down a tree, only coming back up if we don't find a result. In the breadth-first search, we want to check each parent node's child nodes. These two algorithms are very different ways of searching. I found that implementing a depth-first search was easier for me, but I think it is because I was more familiar with this concept. The $FindMinimum$ function was highly challenging and would have taken me a lot longer. Still, the TA's review helped me understand how to use the priority queue to determine how to search for the minimum value in the matrix.

Results

The results all came out as they should. I ran the file and piped it into a results.txt file. The file will be submitted with my source code for examination.

Conclusion

In conclusion, this project was very successful. In the end, I achieved my desired results. I also refreshed my knowledge of basic python skills, learned some of the NumPy modules, and refreshed my searching algorithms knowledge. Overall I am very happy with the results and look forward to attempting the next project.