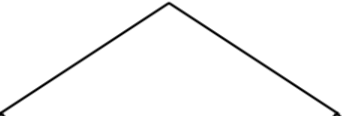
**Lab 1 Report**

The problems we are trying to solve is to perform recursion to draw certain shapes over and over to create a particular figure. For the first set of problems a, b, c for squares I took the program that was given to us by the professor and began to modify the w or the weighting and the starting point in the program. This is so that the first square starts in the same place as the first square drawn in lab 1 A. I then counted the number of squares in Lab 1 A and changed the n to that number to draw the same number of squares as the lab. Now my program and the lab looked the same. So I then repeated this process for b and c. For the second set for a, b, c for circles I took the approach of taking the circles program modifying it by decreasing the number of times it was called to the number of circles which was 3. I then altered the weight until the separation of the circles were the same. I repeated these two processes for b and c.

For number one the problem was to draw the figure. I broke it down into first drawing figure A by starting with drawing the big square. I next focused on drawing each smaller square which is half the size of the big square and each in the corner of big square with the midpoint of the smaller squares the exact point of the big square. Then I drew the bottom left square and I did this by taking the original square dividing the size by 2 and then subtracting 400 from each x and y coordinate to shift the square to the bottom left corner. I then drew the top right square which I divided the size by 2 of the big square and then added by 400 to each x and y coordinate to shift it to the top right corner. I next did the top left corner in which I drew the top left square by taking half the size of the big square and subtracting 400 from its x coordinates only to move it left and adding 400 to its y coordinates to then only shift it up. I then drew the bottom right square by taking half the size of the big square and adding 400 from its x coordinates only to move it right and subtracting 400 to its y coordinates only to shift it down. I then recursively called each small square to draw until n = 0 and subtract one from n each time.

For number 2 the approach was to first get the center to shift left by dividing by a smaller factor each time so that the center shifts left but never outside of the original circle. The next step was to make the radius smaller and smaller each time and to do this we divide by a certain factor.

For number 3 we first plot this figure.



Then we focus on creating the left side of this exact figure but half the size, shifted down, and to the left. In order to do this first divide all coordinates by 2 to make it small next we subtract all the x and y coordinates by the original size to shift it left and down. We then focus on the right side to make it smaller to do this we first divide all coordinates by 2 to make it small then we subtract all the x and y coordinates by the original size to shift it left and down. Following this we recursively call left and right n -1 times to produce this figure n-1 times.

For number 4 it is very similar to number one except this time it is circles and the four smaller circles are drawn in the center, left, right, top, and bottom of the center circle instead of the corners. The first step is to get the smaller circles' radius. They are the same for each smaller circle. The radius of the smaller circles is the big circle's radius divided by 3 because 3 \* radius is the smaller circle's radius which is equal to the radius of the big circle. You then draw the center circle and when doing so no shift is necessary and also use the smaller radius. Then you draw the left circle by moving the center by radius \* 2 distance and to do this you subtract the center x values by the radius \* 2 values. You draw the right circle by adding the same distance to the x values only. To draw the top circle you have to add the same distance to the y values only. To draw the bottom circle you have to  subtract by the same distance to the y values only. You recursively call each circle top ,bottom, right, left and top and it will now draw the figures.

**Experiments**

**ABC square**

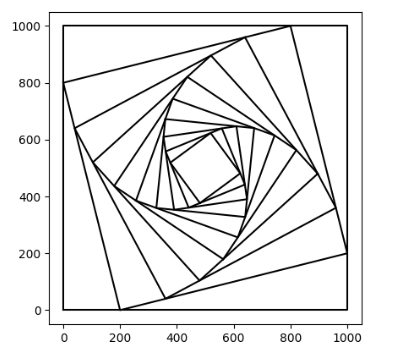
Input

orig\_size = 1000

p = np.array([[0,0],[0,orig\_size],[orig\_size,orig\_size],[orig\_size,0],[0,0]])

draw\_squares(ax,10,p,.2)

Output



Running Time

T(n)=T(n-1) + 1

T(1)=1

T(2)=2

T(3)=3

T(4)=4

O(n)

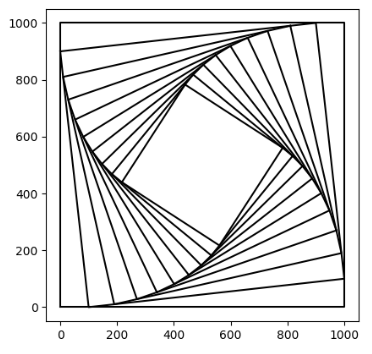
Input

orig\_size = 1000

p = np.array([[0,0],[0,orig\_size],[orig\_size,orig\_size],[orig\_size,0],[0,0]])

draw\_squares(ax,10,p,.1)

Output



Running Time

T(1)=1

T(2)=2

T(3)=3

T(4)=4

O(n)

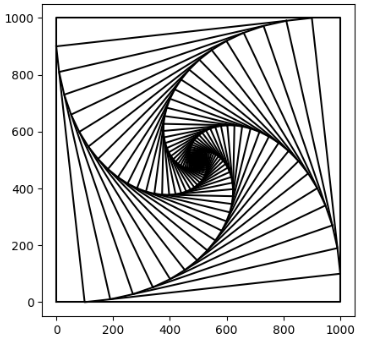
Input

orig\_size = 1000

p = np.array([[0,0],[0,orig\_size],[orig\_size,orig\_size],[orig\_size,0],[0,0]])

draw\_squares(ax,150,p,.1)

Output



Running Time

T(1)=1

T(2)=2

T(3)=3

T(4)=4

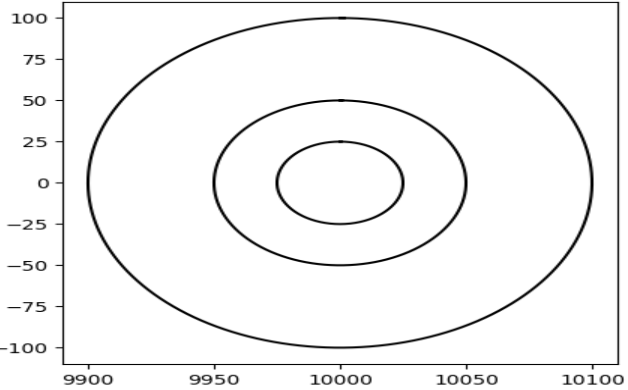
O(n)

**ABC Circles**

Input

draw\_circles(ax, 3, [10000,0], 100,.5)

Output



Running Time

T(1)=1

T(2)=2

T(3)=3

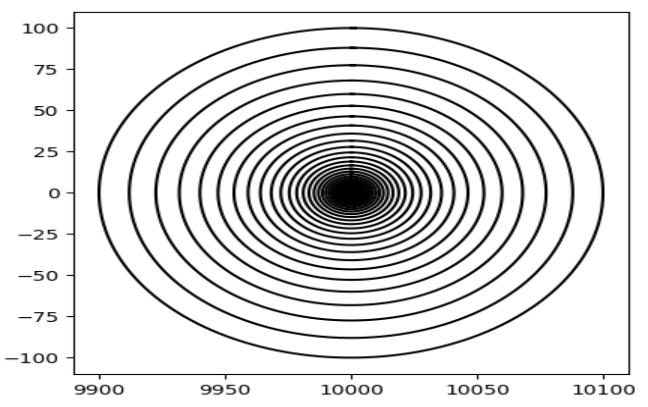
T(4)=4

O(n)

Input

draw\_circles(ax, 400, [10000,0], 100,.88)

Output



Running Time

T(1)=1

T(2)=2

T(3)=3

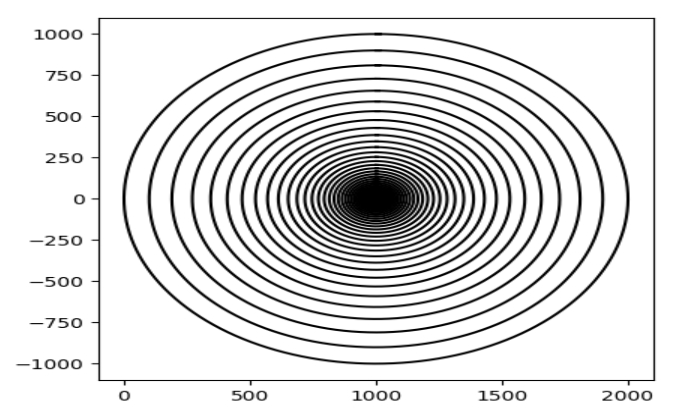
T(4)=4

O(n)

Input

draw\_circles(ax, 700, [1000,0], 1000,.9)

Output



Running Time

T(1)=1

T(2)=2

T(3)=3

T(4)=4

O(n)

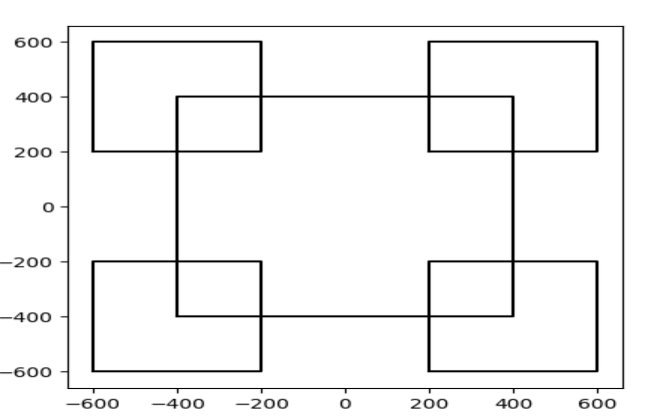
**1. Corner squares**

Input

S=800

draw\_squares(ax,2,array\_pts)

Output



Running Time

T(n)=4T(n-1)+1

T(1)=1

T(2)=5

T(3)=21

T(4)=85

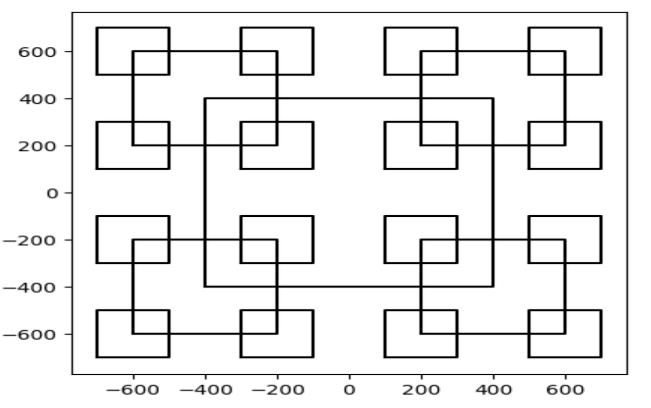
O(n^2)

Input

S=800

draw\_squares(ax,3,array\_pts)

Output



Running Time

T(n)=4T(n-1)+1

T(1)=1

T(2)=5

T(3)=21

T(4)=85

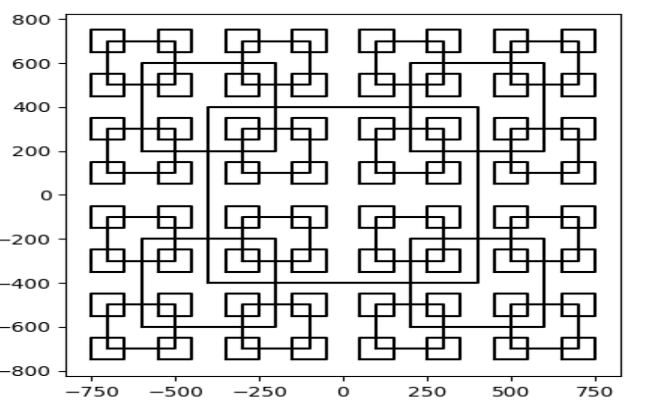
O(n^3)

Input

S=800

draw\_squares(ax,4,array\_pts)

Output



Running Time

T(n)=4T(n-1)+1

T(1)=1

T(2)=5

T(3)=21

T(4)=85

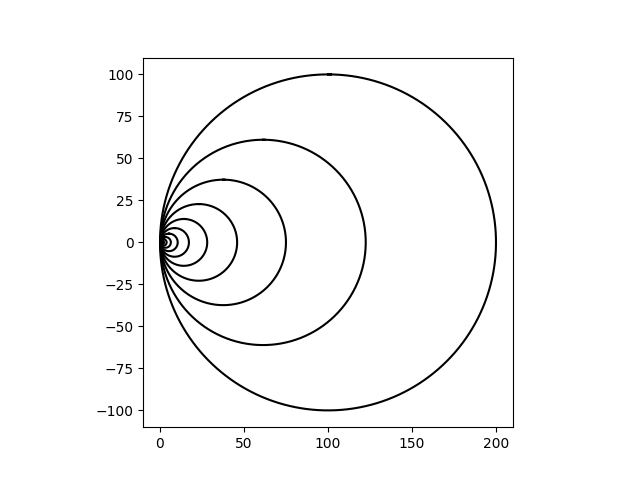
O(n^3)

**2.Circles**

Input

draw\_circles(ax, 50, [100,0], 100,.99)

Output



Running Time

T(n-1)+1

T(1)=1

T(2)=2

T(3)=3

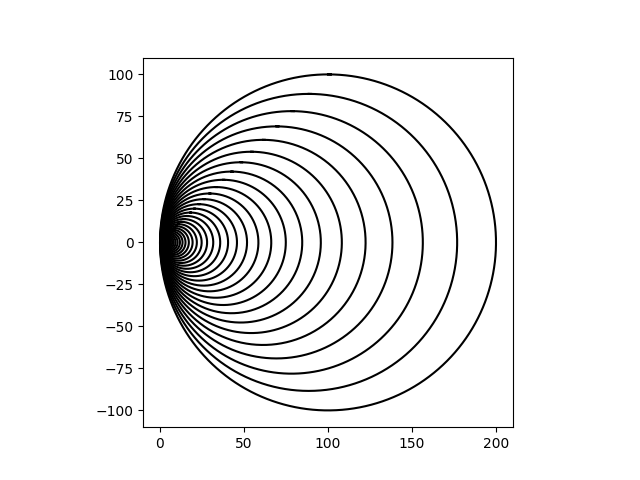
T(4)=4

O(n)

Input

draw\_circles(ax, 50, [100,0], 100,.99)

Output



Running Time

T(n-1)+1

T(1)=1

T(2)=2

T(3)=3

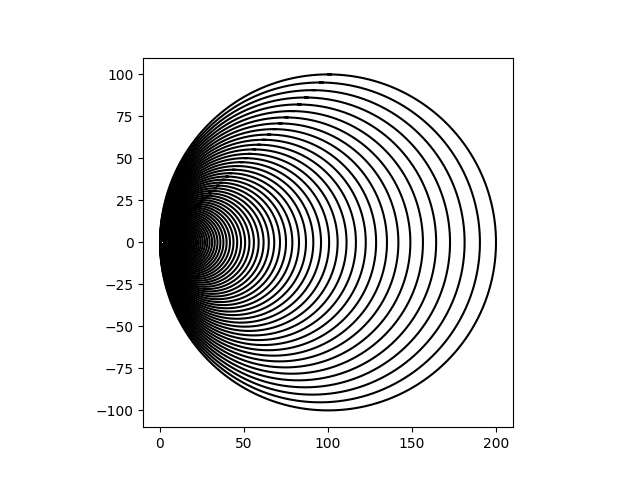
T(4)=4

O(n)

Input

draw\_circles(ax, 90, [100,0], 100,.99)

Output



Running Time

T(n-1)+1

T(1)=1

T(2)=2

T(3)=3

T(4)=4

O(n)

**3 Brackets**

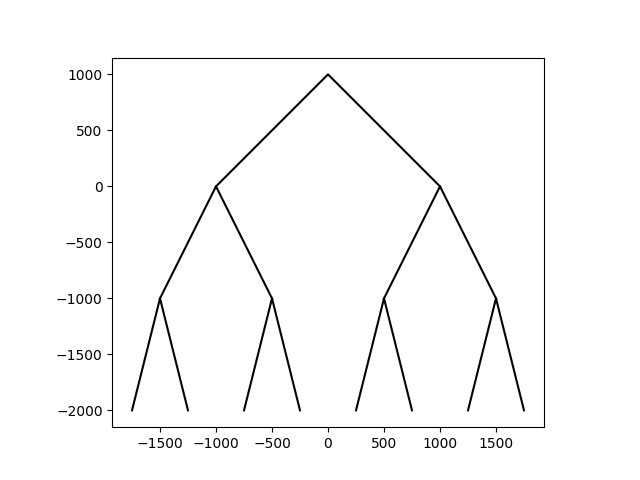
Input

orig\_size = 1000

p = np.array([[-orig\_size,0],[0,orig\_size],[orig\_size,0]])

draw\_squares(ax,3,p)

Output



Running Time

T(n)=2T(n-1)+1

T(1)=1

T(2)=3

T(3)=7

T(4)=15

**O(2^n)**

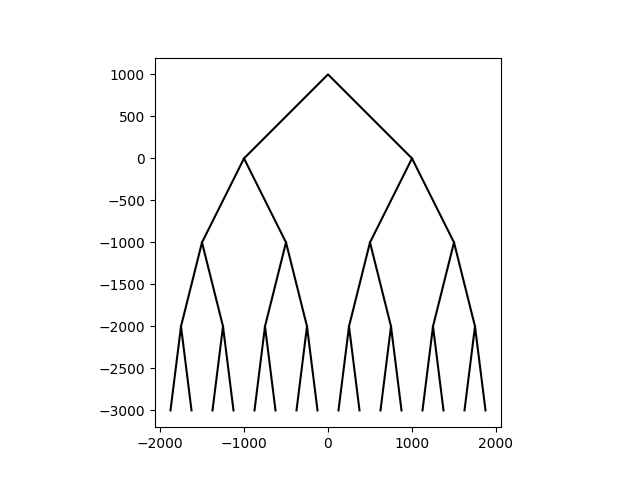
Input

orig\_size = 1000

p = np.array([[-orig\_size,0],[0,orig\_size],[orig\_size,0]])

draw\_squares(ax,4,p)

Output



Running Time

T(n)=2T(n-1)+1

T(1)=1

T(2)=3

T(3)=7

T(4)=15

**O(2^n)**

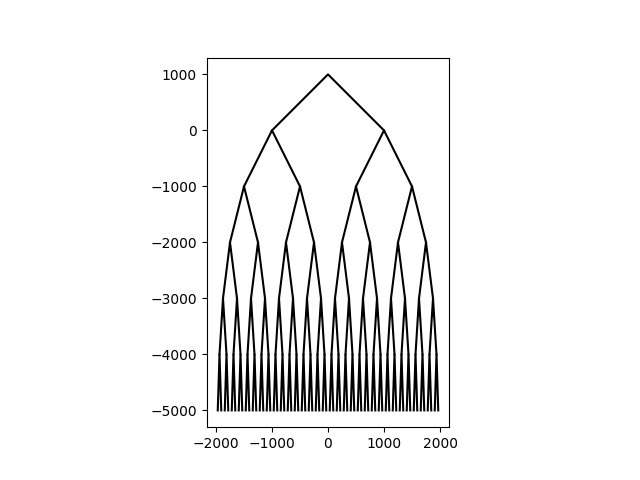
Input

orig\_size = 1000

p = np.array([[-orig\_size,0],[0,orig\_size],[orig\_size,0]])

draw\_squares(ax,6,p)

Output



Running Time

T(n)=2T(n-1)+1

T(1)=1

T(2)=3

T(3)=7

T(4)=15

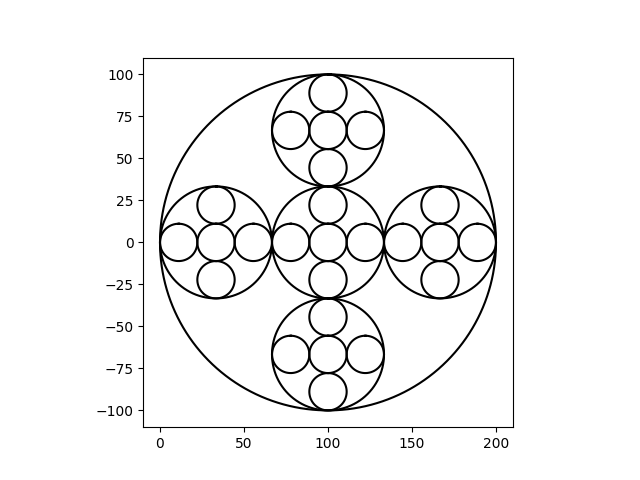
**O(2^n)**

**4. Circles**

Input

draw\_circles(ax, 3, [100,0], 100,.9)

Output



Running Time

T(n)=5T(n-1)+1

T(1)=1

T(2)=6

T(3)=31

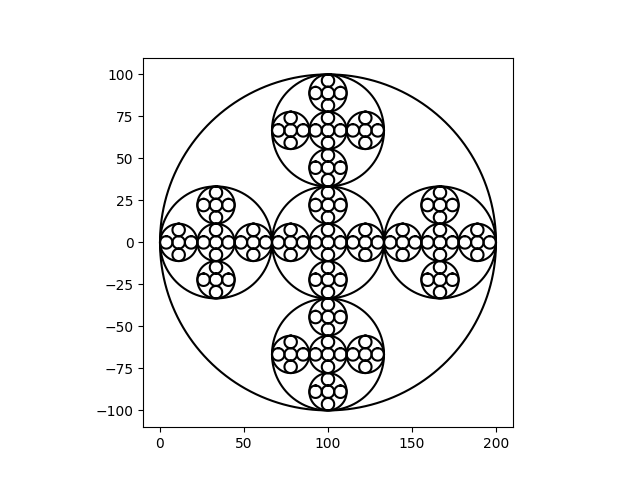
T(4)=(156)

O(n^3)

Input

draw\_circles(ax, 4, [100,0], 100,.9)

Output



Running Time

T(n)=5T(n-1)+1

T(1)=1

T(2)=6

T(3)=31

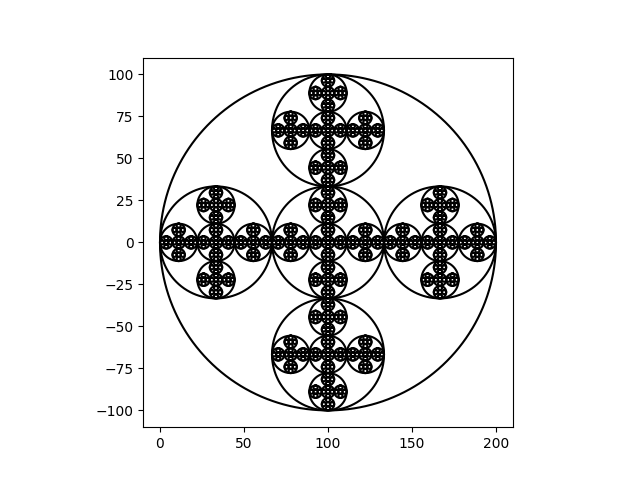
T(4)=(156)

O(n^3)

Input

draw\_circles(ax, 5, [100,0], 100,.9)

Output



Running Time

T(n)=5T(n-1)+1

T(1)=1

T(2)=6

T(3)=31

T(4)=(156)

O(n^3)

**Conclusion**

I have learned how to effectively break down a problem into a smaller problems, solve the smaller problems, then put them together to solve the main problem. I have also learned how to analyze the Big O notation of recursive functions as well as how to use some of numpy in python.

**Appendix**

Source code is in the following links:

<https://github.com/JustinDRui/CS2302-LAB-1.git>

git@github.com:JustinDRui/CS2302-LAB-1.git