Laila Romero

CS 2302 Data Structures

February 12, 2019

Lab 1

* Introduction

For this lab we are trying to create patterns of objects through recursion and algorithms.

Introduction – Description of the problem you are trying to solve

• Proposed Solution Design and Implementation

For the first recursive method, I noticed that the four squares on the corners of the center square were exactly 1/4th the size of the original square. I also noticed that the centers of the smaller squares were the vertices of the larger square. So, in order to get the smaller squares into their respective places I changed their sizes to be 1/4th of the original larger square then moved their center points to the distance of the radius of the larger square.

For the second recursive method, I noticed that the center of the circle was shifted to the right and that was what was creating the pattern. I changed the x value for the center of the circle to be the original x value + the value of the radius so that it would shift the entire circle.

For the third recursive method, I started my center point at the top of the tree and then divided the workspace by what I called “levels” so that every tree would have the same height. For example, if there were three recurrences of the tree such as in the “a” part of the problem, I set the levels to 3 and then set the workspace to be divided by the levels so that the dimensions for each level would be the same. After setting the levels, I would plot each point for the smaller trees at the end vertices of the tree in the level above and make it 1/3 of its dimensions.

For the fourth recursive method, I noticed that the larger circle was able to fit 3 smaller circles across its diameter both horizontally and vertically. I first started by creating the original large circle then creating the smaller center circle to have a diameter that was 1/3rd of the larger diameter and gave it the same center point as the larger circle. I then moved on small leftmost circle and again made it 1/3rd of the original diameter and plotted its center point to be 1/4th of the diameter’s length to the left. I continued this trend for the rightmost circle and the top and bottom.

• Experimental results

For this lab I used a lot of trial and error to try to understand what exactly changed the figures and how each piece of code affected the points and lines. The problem that gave me the most issues was the last circle recursion example. I was able to get the large circle and fit the 5 smaller circles in their respective places but in doing this I also created 5 large circles which shifted along with the smaller circles. I manipulated the formulas I used and still could not get the larger circles to disappear. Aside from this problem, I was able to recreate most of the other images by creating formulas from the patterns and trends I saw such as the squares being 1/4th the size of the center larger square, or the trees being 1/(level size) the dimensions of the tree in the level above it.

In the end my codes were able to reproduce the images for all the original functions (part “a” of each problem) but when it came to the next recursive call it would draw the new image on top of the original image instead of where I had wanted it to. The first circle recursive method did however work for all the parts and was able to work with the recursive methods.

• Conclusion

In this project I learned how to use the basics of matplotlib and the basics of recursion. By using this I learned to plot points, make lines and figures where I wanted them to be. I learned how the basics of recursion worked and how to code these methods using Python. I learned how to analyze out pattern and derive formulas for how to recreate them to the best of my ability.

• Appendix – Source codes

**1) Nested Squares**

the squares use the vertices of the larger square as center points which can be

achieved by dividing the size of the original square by 4. Then use this as the

new center point for the smaller squares and manipulate the x and y coordinates

to align them on the corners

"""

import numpy as np

import matplotlib.pyplot as plt

def draw\_nest(ax,n, x, point):

if n>0:

ax.plot(u[:,0],u[:,1],color='k')

ax.plot(v[:,0],v[:,1],color='k')

ax.plot(w[:,0],w[:,1],color='k')

ax.plot(x[:,0],x[:,1],color='k')

ax.plot(y[:,0],y[:,1],color='k')

draw\_nest(ax,n-1, u, point)

draw\_nest(ax,n-1, v, point)

draw\_nest(ax,n-1, w, point)

draw\_nest(ax,n-1, x, point)

draw\_nest(ax,n-1, y, point)

plt.close("all")

size = 100

point = size/4

n=1

u = np.array([[0,0],[0,size],[size,size],[size,0],[0,0]])

v = np.array([[point\*3,-point],[point\*3,point],[size+point,point],[size+point,-point],[point\*3,-point]])

w = np.array([[-point,-point],[-point,point],[point,point],[point,-point],[-point,-point]])

x = np.array([[-point, size+point],[point, size+point],[point, size-point],[-point,size-point], [-point, size+point]])

y = np.array([[size-point, size+point],[size+point, size+point],[size+point, size-point],[size-point,size-point], [size-point, size+point]])

fig, ax = plt.subplots()

draw\_nest(ax,n, u, point)

draw\_nest(ax,n, v , point)

draw\_nest(ax,n, w , point)

draw\_nest(ax,n, x, point)

draw\_nest(ax,n, y, point)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('squares.png')

**2) Shifted Circles**

to get the circles to the left of the original circle you adjust the center of

each circle after each recursive call by adding the radius to the x coordinate

of the center.

"""

import matplotlib.pyplot as plt

import numpy as np

import math

def circle(center,rad):

n = int(4\*rad\*math.pi)

t = np.linspace(0,6.3,n)

x = center[0]+rad\*np.sin(t)

y = center[1]+rad\*np.cos(t)

return x,y

def draw\_circles(ax,n,center,radius,w):

if n>0:

x,y = circle(center,radius)

ax.plot(x + radius,y,color='k') # will change the center of the new circles

draw\_circles(ax,n-1,center,radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax, 10, [100,0], 100,.6)

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')

**3) Trees**

for the tree example you start at the origin at the top of the tree then

set the other coordinates to half of the radius of the space. You do this for

every level then for the x coordinates and for the y coordinate you take the

number of levels and divide it by the height of the workspace

"""

import numpy as np

import matplotlib.pyplot as plt

def draw\_nest(ax,n, x, size,levels):

if n>0:

ax.plot(p[:,0],p[:,1],color='k')

ax.plot(q[:,0],q[:,1],color='k')

ax.plot(r[:,0],r[:,1],color='k')

ax.plot(s[:,0],s[:,1],color='k')

ax.plot(t[:,0],t[:,1],color='k')

ax.plot(u[:,0],u[:,1],color='k')

ax.plot(v[:,0],v[:,1],color='k')

draw\_nest(ax,n-1,p,size,levels)

plt.close("all")

size = 90

n=1

levels=3

new\_size= size/levels

fig, ax= plt.subplots()

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

q = np.array([[-size/2,-2\*new\_size],[-new\_size,-new\_size],[-new\_size/2,-2\*new\_size]])

r = np.array([[new\_size/2,-2\*new\_size],[new\_size,-new\_size],[new\_size/2+new\_size,-2\*new\_size]])

s = np.array([[new\_size/4,-size], [new\_size/2,-2\*new\_size],[new\_size-new\_size/4,-size]])

t = np.array([[-new\_size/4,-size], [-new\_size/2,-2\*new\_size],[-new\_size+new\_size/4,-size]])

u = np.array([[-2\*new\_size+ new\_size/4,-size], [-new\_size-new\_size/2,-2\*new\_size], [-new\_size-new\_size/4,-size]])

v = np.array([[new\_size+new\_size/4,-size], [new\_size+new\_size/2, -2\*new\_size], [2\*new\_size-new\_size/4,-size]])

draw\_nest(ax,n,p,size,levels)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('squares.png')

**4) Nested Circles**

for the multiple circles you take the center of the main circle then divide the

diameter by 3 to get the new centers for the circles inside of it. then call

recursive method

"""

import matplotlib.pyplot as plt

import numpy as np

import math

def circle(center,rad):

n = int(4\*rad\*math.pi)

t = np.linspace(0,6.3,n)

x = center[0]+rad\*np.sin(t)

y = center[1]+rad\*np.cos(t)

return x,y

def draw\_circles(ax,n,center,radius):

if n>0:

x,y = circle(center,radius)

ax.plot(x,y,color='k')

draw\_circles(ax,n-1,center,radius)

draw\_circles(ax,n-2, center, radius/3)

plt.close("all")

fig, ax = plt.subplots()

radius=100

center= [0,0]

draw\_circles(ax, 3, center, radius)

draw\_circles(ax, 3, [-2\*radius/3,0], radius)

draw\_circles(ax, 3, [2\*radius/3,0], radius)

draw\_circles(ax, 3, [0,2\*radius/3], radius)

draw\_circles(ax, 3, [0,-2\*radius/3], radius)

ax.set\_aspect(1.0)

#ax.axis('off')

plt.show()

fig.savefig('circles.png')