Michael Gonzalez

CS 2302 Data Structures

Olac Fuentes

02/06/2019

Lab 1 – Recursive Figures

For this lab, we have been tasked with replicating various graphs through recursion. We had four different graphs that we had to replicate. Part 1 was to draw squares on each corner of an existing square. Part 2 was to draw circles slowly decreasing in size and moving towards the left. Part 3 was to draw what resembled a binary tree. Part 4 was to draw five circles inside an existing circle every time there was a circle. The biggest challenge that I could see with this assignment is that I would have to figure out the path that my code to go through before coding my solution.

My solution for part 1 was to create a method that used a figure, an “n” value, an “x” coordinate, a “y” coordinate, and a radius value “r.” Inside the method, four variables will be created with the values for each corner points on a square based on the distance between its center point and radius. Next, I would store these four values in an array labeled “p” and plot it on the figure. After the drawing is complete there will be four recursive calls; each call will use one of the four corner values and subtract one from n, this will continue until the “n” value has reached zero.

My solution for part 2 was to create a method that used a figure, an “n” value, an “x” coordinate, a “y” coordinate, a radius value “r,” and a size variable “w.” Inside the method using the x and y values, it will create and plot a circle trough the “circle” method with a radius of “r.” Next the recursive call would do this process again except that its variables have been modified, the x value would be subtracted from the value of (radius\*(1-size)) and the radius would also be modified by multiplying said radius with the size variable (radius\*size) and would subtract 1 from the “n” value until it had reached zero.

My solution for part 3 was to create a method that used a figure, an “n” value, an “x” coordinate, a “y” coordinate, a “p” array with coordinates, and a radius value “r.” Inside the method using the “p” array, it will plot the coordinates in it. Next, it will create variables to hold the new coordinates for the branches. Next, it will update the “p” array with the left branch coordinates and make a recursive call using the updated “p” array, the left “x” coordinate, the updated “y” value, and half the radius. Once the left calls have been made the “p” array will update once more using the right coordinates and will make the same recursive call as before only his time using the right “x” coordinate. Each call will also reduce “n” by 1 and will keep repeating until “n” has reached zero.

My solution for part 4 was to create a method that used a figure, an “n” value, an “x” coordinate, a “y” coordinate, a radius value “r.” Inside the method using the “x” and “y” values, it will create and plot a circle trough the “circle” method with a radius of “r.” Net it will create 4 variables that will hold the new x and y coordinates for the recursive circles. Next make the recursive calls based on the values from the new center points, dividing the current radius by 3 and decrease “n” by 1.

For part 1, To find how I wanted to code my solution, I took the graphs to the whiteboard and began drawing. I started by drawing one square; then I drew one a square on the corner of the first. I noticed how if I could use coordinates to start my squares I could pass those coordinates through recursion and make new squares. And by using the plot command if I update the array holding the coordinates there would be nothing else to do. So, I ended up breaking my code into three parts. 1. Create new variables that store the new coordinates. 2. Store and plot new square.3. Recursive call new squares based on the specific corner of the existing square. And so, I was able to come up with my solution for drawing squares recursively.

For part 2, To find how I wanted to code my solution, I took the graphs to the whiteboard and began drawing. I drew part a and noticed that it appeared as though each circle started from that leftmost point, but then I realized that the way the circle method plots circles s trough a center point and an equally surrounding radius meaning I saw the graphs wrong. For sure I could tell that the circles were decreasing in size to what I would say is half their previous radius. After this, I noticed how there was no change in the y coordinates of the circles but the x coordinates where moving towards the negative side. Now seeing how each circle came to be, I broke down the code into two parts. 1. Create and plot a circle based on its center point and radius. 2. Recursive call with modified inputs. And so I was able to come up with my solution for drawing circles recursively

For part 3, To find how I wanted to code my solution, I took the graphs to the whiteboard and began drawing. Each time I drew a line, I would write down on the side the code to plot it in python. Then I decided to draw only the left side of the tree and then the right side if there was any, this led me to realize that the endpoint of the line is where my new recursive call should start from line is the code from the first line, and then the stating point would be where the endpoint is. So, I ended up breaking this code into 3 parts. 1.plot line stored in the P array. 2. Create variables to store the new coordinates 3. Recursive call with the left coordinates and then the right coordinate. And so, I was able to come up with my solution for drawing trees recursively.

For part 4, To find how I wanted to code my solution, I took the graphs to the whiteboard and began drawing. First thing would be to draw one big circle. Next, I drew the five individual circles in at shape. I realized that the middle circle is just a reduced version of the first circle and then I began to think that all 5 of the small circles were the same just placed in 5 different x and y coordinates. And that in each circle from then on will have the same reduced five circles; I realized then that I could break down my code to 3 parts. 1. Create and plot a circle. 2. Create variables to store the new coordinates of the reduced circles. 3. Make recursive calls with the new coordinates corresponding to each circle, divide the radius “r” by 3 and decrease “n” by 1.

In conclusion, I learned how to program various applications of recursion ranging from drawing squares shifting in size to creating expansive trees. Aside from recursion I have learned how to utilize the Python numpy and plt libraries. I have become more comfortable with coding in python and I believe that I will be able to learn more from future labs to come.

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

– Michael Gonzalez

Appendix – code

1.

a.

import matplotlib.pyplot as plt

import numpy as np

def draw\_squares\_recursively(ax,n,x,y,r):

if n>0:

# Create 4 variables with coordinates for square corners

xleft =x-r

xrigth =x+r

ytop =y+r

ybot =y-r

#store and plot square

p = np.array([[xleft,ybot],[xleft,ytop],[xrigth,ytop],[xrigth,ybot],[xleft,ybot]])

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

#recursive calls on each square corner

draw\_squares\_recursively(ax,n-1,xleft,ybot,r/2)

draw\_squares\_recursively(ax,n-1,xleft,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ybot,r/2)

plt.close("all")

fig, ax = plt.subplots()

center = 500

draw\_squares\_recursively(ax,2,0,0,500) #Change was done here “n = 2”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('squares.png')

b.

import matplotlib.pyplot as plt

import numpy as np

def draw\_squares\_recursively(ax,n,x,y,r):

if n>0:

# Create 4 variables with coordinates for square corners

xleft =x-r

xrigth =x+r

ytop =y+r

ybot =y-r

#store and plot square

p = np.array([[xleft,ybot],[xleft,ytop],[xrigth,ytop],[xrigth,ybot],[xleft,ybot]])

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

#recursive calls on each square corner

draw\_squares\_recursively(ax,n-1,xleft,ybot,r/2)

draw\_squares\_recursively(ax,n-1,xleft,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ybot,r/2)

plt.close("all")

fig, ax = plt.subplots()

center = 500

draw\_squares\_recursively(ax,3,0,0,500) #Change was done here “n = 3”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('squares.png')

c.

import matplotlib.pyplot as plt

import numpy as np

def draw\_squares\_recursively(ax,n,x,y,r):

if n>0:

# Create 4 variables with coordinates for square corners

xleft =x-r

xrigth =x+r

ytop =y+r

ybot =y-r

#store and plot square

p = np.array([[xleft,ybot],[xleft,ytop],[xrigth,ytop],[xrigth,ybot],[xleft,ybot]])

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

#recursive calls on each square corner

draw\_squares\_recursively(ax,n-1,xleft,ybot,r/2)

draw\_squares\_recursively(ax,n-1,xleft,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ytop,r/2)

draw\_squares\_recursively(ax,n-1,xrigth,ybot,r/2)

plt.close("all")

fig, ax = plt.subplots()

center = 500

draw\_squares\_recursively(ax,4,0,0,500) #Change was done here “n = 4”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('squares.png')

2.

a.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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\_recursively(ax,n,xcenter,ycenter,radius,w):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

#recursive call with modified inputs

draw\_circles\_recursively(ax,n-1,xcenter-radius\*(1-w),ycenter,radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

draw\_circles\_recursively(ax, 10, xcenter,ycenter, 1000,.5)

#Change was done here “n = 10 w = .5”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')

b.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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\_recursively(ax,n,xcenter,ycenter,radius,w):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

#recursive call with modified inputs

draw\_circles\_recursively(ax,n-1,xcenter-radius\*(1-w),ycenter,radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

draw\_circles\_recursively(ax, 50, xcenter,ycenter, 1000,.9)

#Change was done here “n = 50 w = .9”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')

c.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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\_recursively(ax,n,xcenter,ycenter,radius,w):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

#recursive call with modified inputs

draw\_circles\_recursively(ax,n-1,xcenter-radius\*(1-w),ycenter,radius\*w,w)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

draw\_circles\_recursively(ax, 100, xcenter,ycenter, 1000,.95)

#Change was done here “n = 100 w = .95”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')

3. tree\_recursion

a.

import matplotlib.pyplot as plt

import numpy as np

def tree\_recursion(ax,n,x,y,p,r):

if n>0:

#plot line that is stored in "p" array

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

#create variables to store new coordinate values for new branches

xleft = x-r

xrigth = x+r

ydown = y-1

#update "p" array with left coordinates and call method

p = np.array([[x,y],[xleft,ydown]])

tree\_recursion(ax,n-1,xleft,ydown,p,r\*.5)

#update "p" array with right coordinates and call method

p = np.array([[x,y],[xrigth,ydown]])

tree\_recursion(ax,n-1,xrigth,ydown,p,r\*.5)

plt.close("all")

fig, ax = plt.subplots()

x = 0

y = 0

p = np.array([[x,y],[0,0]])

tree\_recursion(ax,4,0,0,p,1) #Change was done here “n = 4”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('trees.png')

b.

import matplotlib.pyplot as plt

import numpy as np

def tree\_recursion(ax,n,x,y,p,r):

if n>0:

#plot line that is stored in "p" array

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

#create variables to store new coordinate values for new branches

xleft = x-r

xrigth = x+r

ydown = y-1

#update "p" array with left coordinates and call method

p = np.array([[x,y],[xleft,ydown]])

tree\_recursion(ax,n-1,xleft,ydown,p,r\*.5)

#update "p" array with right coordinates and call method

p = np.array([[x,y],[xrigth,ydown]])

tree\_recursion(ax,n-1,xrigth,ydown,p,r\*.5)

plt.close("all")

fig, ax = plt.subplots()

x = 0

y = 0

p = np.array([[x,y],[0,0]])

tree\_recursion(ax,5,0,0,p,1) #Change was done here “n = 5”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('trees.png')

c.

import matplotlib.pyplot as plt

import numpy as np

def tree\_recursion(ax,n,x,y,p,r):

if n>0:

#plot line that is stored in "p" array

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

#create variables to store new coordinate values for new branches

xleft = x-r

xrigth = x+r

ydown = y-1

#update "p" array with left coordinates and call method

p = np.array([[x,y],[xleft,ydown]])

tree\_recursion(ax,n-1,xleft,ydown,p,r\*.5)

#update "p" array with right coordinates and call method

p = np.array([[x,y],[xrigth,ydown]])

tree\_recursion(ax,n-1,xrigth,ydown,p,r\*.5)

plt.close("all")

fig, ax = plt.subplots()

x = 0

y = 0

p = np.array([[x,y],[0,0]])

tree\_recursion(ax,7,0,0,p,1.5) #Change was done here “n = 7 & r = 1.5”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('trees.png')

4.

a.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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 circle\_in\_circle(ax,n,xcenter,ycenter,radius):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

# Create 4 variables with new circle coordinates

xleft = xcenter + (radius\*(2/3))

xrigth = xcenter - (radius\*(2/3))

ytop = ycenter + (radius\*(2/3))

ybot = ycenter - (radius\*(2/3))

#reduce radius by 3

radius = radius/3

#recursive calls for the 5 circles inside the first circle

circle\_in\_circle(ax,n-1,xcenter,ycenter,radius)

circle\_in\_circle(ax,n-1,xleft,ycenter,radius)

circle\_in\_circle(ax,n-1,xrigth,ycenter,radius)

circle\_in\_circle(ax,n-1,xcenter,ytop,radius)

circle\_in\_circle(ax,n-1,xcenter,ybot,radius)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

circle\_in\_circle(ax, 3, xcenter,ycenter, 1000) #Change is done here “n = 3”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles\_in\_circles.png')

b.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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 circle\_in\_circle(ax,n,xcenter,ycenter,radius):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

# Create 4 variables with new circle coordinates

xleft = xcenter + (radius\*(2/3))

xrigth = xcenter - (radius\*(2/3))

ytop = ycenter + (radius\*(2/3))

ybot = ycenter - (radius\*(2/3))

#reduce radius by 3

radius = radius/3

#recursive calls for the 5 circles inside the first circle

circle\_in\_circle(ax,n-1,xcenter,ycenter,radius)

circle\_in\_circle(ax,n-1,xleft,ycenter,radius)

circle\_in\_circle(ax,n-1,xrigth,ycenter,radius)

circle\_in\_circle(ax,n-1,xcenter,ytop,radius)

circle\_in\_circle(ax,n-1,xcenter,ybot,radius)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

circle\_in\_circle(ax, 4, xcenter,ycenter, 1000) #Change is done here “n = 4”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles\_in\_circles.png')

c.

import matplotlib.pyplot as plt

import numpy as np

import math

#method to create a circle

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 circle\_in\_circle(ax,n,xcenter,ycenter,radius):

if n>0:

#Create and plot a circle based on its center point and radius

center = [xcenter,ycenter]

x,y = circle(center,radius)

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

# Create 4 variables with new circle coordinates

xleft = xcenter + (radius\*(2/3))

xrigth = xcenter - (radius\*(2/3))

ytop = ycenter + (radius\*(2/3))

ybot = ycenter - (radius\*(2/3))

#reduce radius by 3

radius = radius/3

#recursive calls for the 5 circles inside the first circle

circle\_in\_circle(ax,n-1,xcenter,ycenter,radius)

circle\_in\_circle(ax,n-1,xleft,ycenter,radius)

circle\_in\_circle(ax,n-1,xrigth,ycenter,radius)

circle\_in\_circle(ax,n-1,xcenter,ytop,radius)

circle\_in\_circle(ax,n-1,xcenter,ybot,radius)

plt.close("all")

fig, ax = plt.subplots()

xcenter = 0

ycenter = 0

circle\_in\_circle(ax, 5, xcenter,ycenter, 1000) #Change is done here “n = 5”

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles\_in\_circles.png')