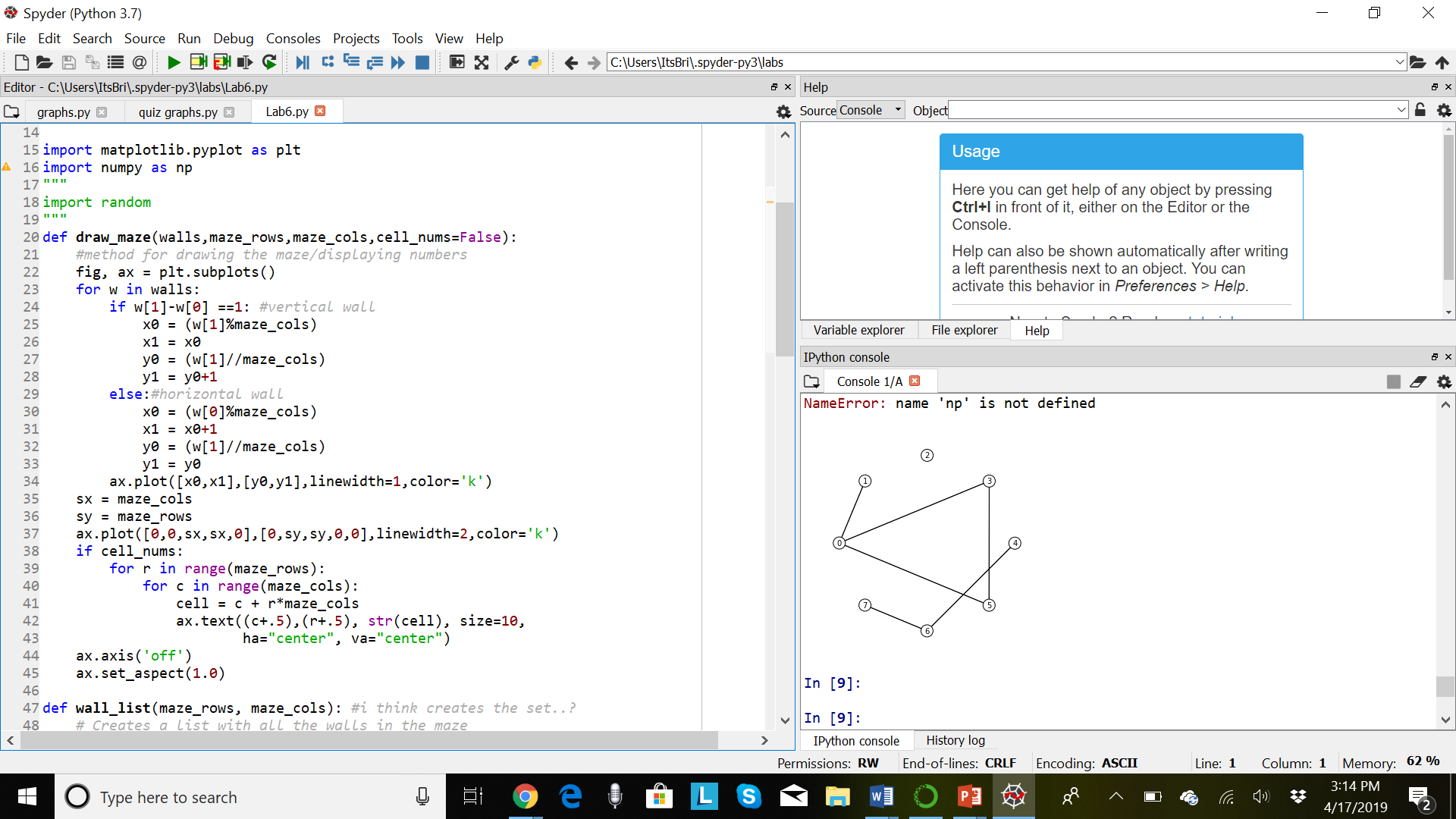
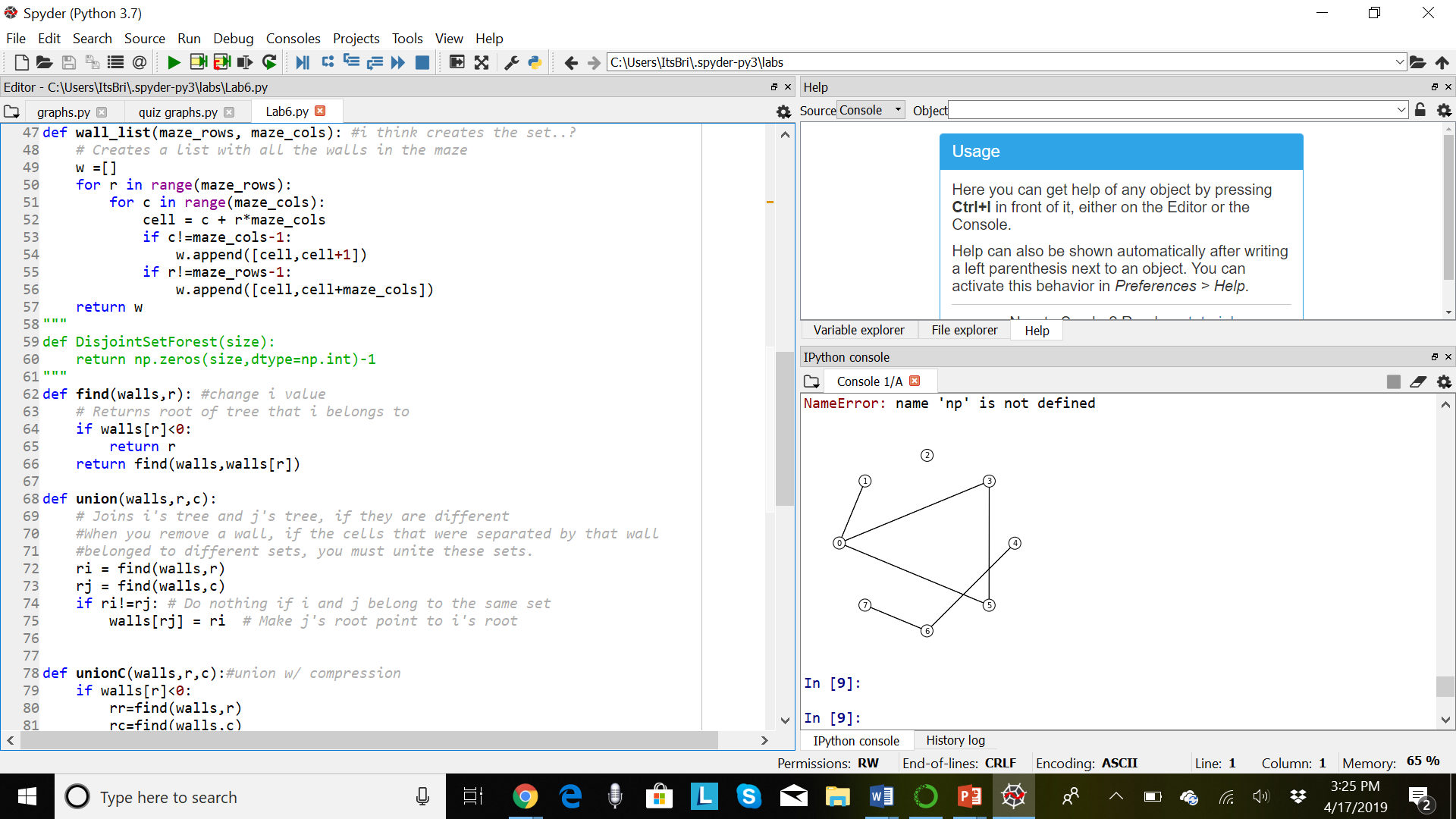
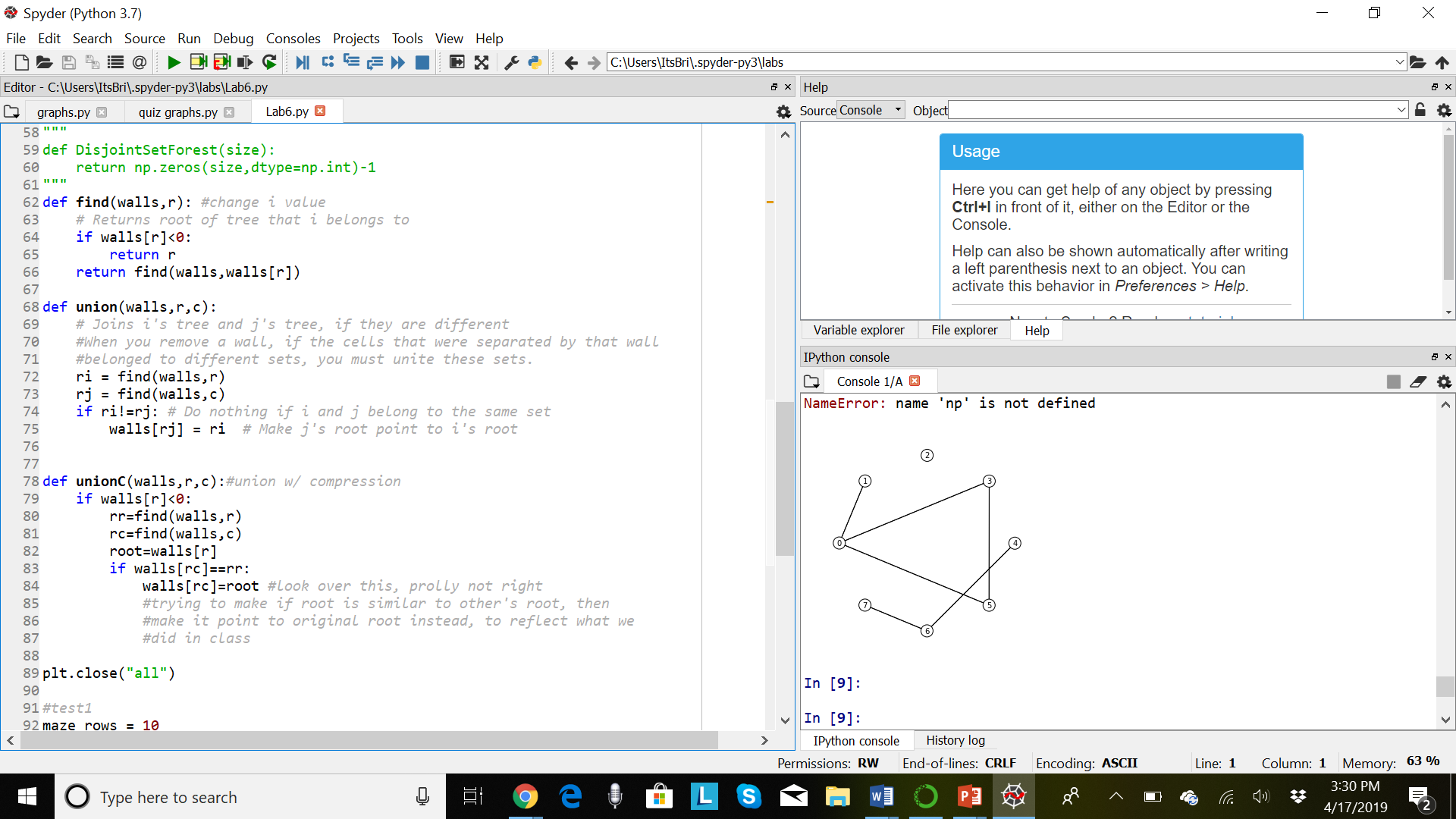
Lab 6 Report

For this lab, I couldn’t fully understand this concept of DSF, disjoint set forests. At the beginning of my code, I first added the given code from Dr.Fuentes. 

After reviewing how this code works to draw the maze, I started implementing my own method to using the given method of, “wall\_list”. I used methods, modifying them, to use what, “wall\_list”, was given.



I managed to finish my method of union standard, but as for union with compression, I was having trouble understanding how to code this. Because of this, I was stuck on this, and didn’t realize that one of the given methods in DSF, had this method finished. Over time, I slowly began to understand how to use the DSF for creating the maze, but ran out of time.



Honesty Statement:

Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve

copying from or providing information to another student, possessing unauthorized materials during a test, or

falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone

represents the work or ideas of another person as his/her own. Collusion involves collaborating with another

person to commit an academically dishonest act. Professors are required to - and will - report academic

dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

I hereby state hereby this code is mine and mine alone.

Appendix:

import matplotlib.pyplot as plt

import numpy as np

"""

import random

"""

def draw\_maze(walls,maze\_rows,maze\_cols,cell\_nums=False):

#method for drawing the maze/displaying numbers

fig, ax = plt.subplots()

for w in walls:

if w[1]-w[0] ==1: #vertical wall

x0 = (w[1]%maze\_cols)

x1 = x0

y0 = (w[1]//maze\_cols)

y1 = y0+1

else:#horizontal wall

x0 = (w[0]%maze\_cols)

x1 = x0+1

y0 = (w[1]//maze\_cols)

y1 = y0

ax.plot([x0,x1],[y0,y1],linewidth=1,color='k')

sx = maze\_cols

sy = maze\_rows

ax.plot([0,0,sx,sx,0],[0,sy,sy,0,0],linewidth=2,color='k')

if cell\_nums:

for r in range(maze\_rows):

for c in range(maze\_cols):

cell = c + r\*maze\_cols

ax.text((c+.5),(r+.5), str(cell), size=10,

ha="center", va="center")

ax.axis('off')

ax.set\_aspect(1.0)

def wall\_list(maze\_rows, maze\_cols): #i think creates the set..?

# Creates a list with all the walls in the maze

w =[]

for r in range(maze\_rows):

for c in range(maze\_cols):

cell = c + r\*maze\_cols

if c!=maze\_cols-1:

w.append([cell,cell+1])

if r!=maze\_rows-1:

w.append([cell,cell+maze\_cols])

return w

"""

def DisjointSetForest(size):

return np.zeros(size,dtype=np.int)-1

"""

def find(walls,r): #change i value

# Returns root of tree that i belongs to

if walls[r]<0:

return r

return find(walls,walls[r])

def union(walls,r,c):

# Joins i's tree and j's tree, if they are different

#When you remove a wall, if the cells that were separated by that wall

#belonged to different sets, you must unite these sets.

ri = find(walls,r)

rj = find(walls,c)

if ri!=rj: # Do nothing if i and j belong to the same set

walls[rj] = ri # Make j's root point to i's root

def unionC(walls,r,c):#union w/ compression

if walls[r]<0:

rr=find(walls,r)

rc=find(walls,c)

root=walls[r]

if walls[rc]==rr:

walls[rc]=root #look over this, prolly not right

#trying to make if root is similar to other's root, then

#make it point to original root instead, to reflect what we

#did in class

plt.close("all")

#test1

maze\_rows = 10

maze\_cols = 15

"""

#test2

maze\_rows = 5

maze\_cols = 10

"""

"""

#test3

maze\_rows = 0

maze\_cols = 2

"""

"""

#test4

maze\_rows = 1

maze\_cols = 5

"""

walls = wall\_list(maze\_rows,maze\_cols)

"""

union(walls,2,4)

union(walls,7,9)

"""

draw\_maze(walls,maze\_rows,maze\_cols,cell\_nums=True) #prints figure of numbered maze

"""

#randomly removing walls

for i in range(len(walls)//2): #Remove 1/2 of the walls

d = random.randint(0,len(walls)-1)

print('removing wall ',walls[d])

walls.pop(d)

"""

draw\_maze(walls,maze\_rows,maze\_cols) #prints maze w/out numbers