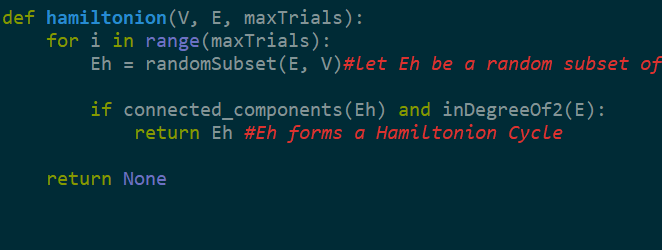
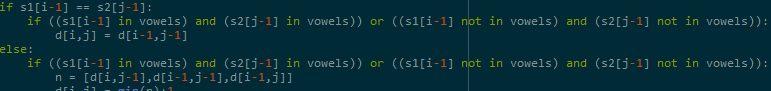
**Lab07 Report**

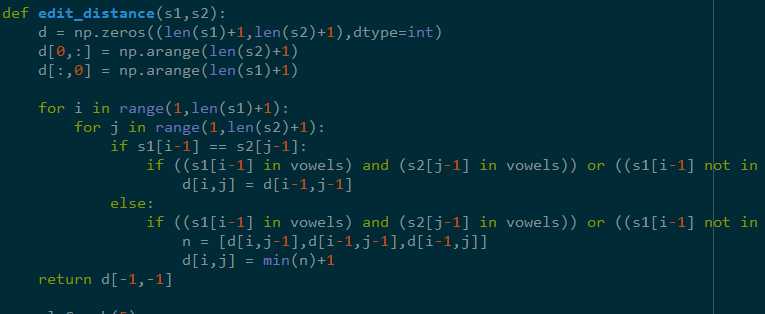
In lab 7 we are asked to implement a function to solve the Hamiltonian Cycle problem. The first thing that needs to be done if a for loop to iterate through max trails. Inside of the loop a variable ‘Eh’ is initiated to be a random subset of E of size V being passed in. I then add an if statement to check if connected components function with Eh being passed in and indegree function with E being passed in are both true. if they are then we will return Eh which forms the Hamiltonian cycle. And otherwise we will return None. The following is a screenshot of the function.



The next function we are being asked to implement is the Dynamic Programing, which will be made from the edit distance formula given in class. The function remained very similar to the edit distance formula. The formula will still have a nested for loop through both of the strings ‘s1’ and ‘s2’. Both of them will be compared in the first if statement at i-1 and j-1. Then next step was to add another if statement which checks if both of the s1 and s2 belong in vowels string which contains vowels. This is done in the following.



The following is the entire function



Source Code:

import graph\_AL as al

import dsf

from random import randint

import numpy as np

vowels = ["a","e","i","o","u"]

#returns true of g has 1 connected component and the in degree of every vertex in V is 2

def connected\_components(g):

vertices = len(g.al)

components = vertices

s = dsf.DSF(vertices )

for v in range(vertices):

for edge in g.al[v]:

components -= s.union(v,edge.dest)

s.draw()

if components > 1:

return True

return False

#returns a random subset of g of size V

def randomSubset(g, V):

` for i in range(len(g.al) - V):

g.al.remove(randint(0, V)) #remove two random vertices

return g

#Check if all nodes have in degree of 2

def inDegreeOf2(g):

for i in range(len(g.al)):

count = 0

for j in range(g.al[i]):

if g.al[j].dest == i:

count += 1

if count != 2:

return False

return True

def hamiltonion(V, E, maxTrials):

for i in range(maxTrials):

Eh = randomSubset(E, V)#let Eh be a random subset of E of size V

if connected\_components(Eh) and inDegreeOf2(E):

return Eh #Eh forms a Hamiltonion Cycle

return None

def edit\_distance(s1,s2):

d = np.zeros((len(s1)+1,len(s2)+1),dtype=int)

d[0,:] = np.arange(len(s2)+1)

d[:,0] = np.arange(len(s1)+1)

for i in range(1,len(s1)+1):

for j in range(1,len(s2)+1):

if s1[i-1] == s2[j-1]:

if ((s1[i-1] in vowels) and (s2[j-1] in vowels)) or ((s1[i-1] not in vowels) and (s2[j-1] not in vowels)):

d[i,j] = d[i-1,j-1]

else:

if ((s1[i-1] in vowels) and (s2[j-1] in vowels)) or ((s1[i-1] not in vowels) and (s2[j-1] not in vowels)):

n = [d[i,j-1],d[i-1,j-1],d[i-1,j]]

d[i,j] = min(n)+1

return d[-1,-1]

g = al.Graph(5)

g.insert\_edge(0,1)

g.insert\_edge(1,2)

g.insert\_edge(2,3)

g.insert\_edge(3,4)

g.insert\_edge(4,0)

g.draw();

hamiltonion(g, 4, 20)

print(edit\_distance('sand', 'sound'))

#inDegreeOf2(g)