import networkx as nx

import matplotlib.pyplot as plt

import random as r

import numpy as np

def add\_edge(g,p):

#p is the probabilisatic model

for u in g.nodes():

for v in g.nodes():

if u!=v:

ran=r.randint(0,1)

if ran<=p:

g.add\_edge(u,v)

else:

continue

return g

def points(n): #Assign the initial points to each nodes here 100

assign=[100 for ele in range(n)]

return assign

def distribute(g,point):

new\_point=[0 for ele in range(len(g))]

prev\_point=point

for var in g.nodes():

print g.nodes()

out=g.out\_edges(var)

if len(out)==0:

new\_point[var]+=prev\_point[var]

else:

share=float(prev\_point[var])/len(out)

for ele in out:

print ele[1]

new\_point[ele[1]]+=share

return new\_point

def distribute\_points(g,point,limit):

prev\_point=point

for ele in range(limit):

new\_point=distribute(g,prev\_point)

prev\_point=new\_point

return new\_point

def sort(point):

pnt=np.array(point)

sorted=np.argsort(-pnt)

return sorted

def check(s,p):

for each in range(len(s)):

if s[each]!=p[each][0]:

return False

return True

n=15

g=nx.DiGraph()

for ele in range(n):

g.add\_node(ele)

g=add\_edge(g,0.3)

point=points(n)

d\_point=distribute\_points(g,point,50)

sorted\_nodes=sort(d\_point)

pr=nx.pagerank(g)

pr\_sorted=sorted(pr.items(),key=lambda x:x[1],reverse=True)

print 'Sorted nodes: using my method'

print sorted\_nodes

print 'Sorted nodes: using inbuilt pagerank'

for i in pr\_sorted:

print i[0],

print '\nsimilarity: ',check(sorted\_nodes,pr\_sorted)