import networkx as nx

import random

import numpy as np

G=nx.DiGraph()

G.add\_edge(‘UUNomor10Tahun1995’,’UUD1945’)

G.add\_edge(‘UUNomor10Tahun1995’,’UUNomor17Tahun2006’)

G.add\_edge(‘UUNomor28Tahun1999’,’UUD 1945’)

G.add\_edge(‘UUNomor36Tahun2008‘,’UUD 1945’)

G.add\_edge(‘UUNomor36Tahun2008’,’UUNomor6Tahun 1983’)

G.add\_edge(‘UUNomor36Tahun2008’,’UUNomor67Tahun 1983’)

G.add\_edge(‘UUNomor26Tahun2009’,’UUNomor41Tahun 2008’)

G.add\_edge(‘UUNomor2Tahun2018’,’UUNomor5Tahun2014’)

# Function to add edges to the directed graph

def add\_edges(G,p):

for i in G.nodes():

for j in G.nodes():

if i != j :

r = random.random()

if r <= p:

G.add\_edge(i,j)

else:

continue

return G

# Function to sort nodes according to their accumulated points

# We will denote "random-walk-point" by "RWP"

def get\_nodes\_sorted\_by\_RWP(RWP\_list):

RWP\_array = np.array(RWP\_list) # Converting list to numpy array

nodes\_sorted\_by\_RWP = np.argsort(-RWP\_array) # The minus sign is used for sorting in descending order

return nodes\_sorted\_by\_RWP

# Function to perform random walks and increment random-walk-point of visited node by 1

def random\_walk(G):

nodes = list(G.nodes()) # List of nodes of graph

RWP = [0 for i in range(G.number\_of\_nodes())] # List containing RWP of all nodes which are initially 0

s\_node = random.choice(nodes) # Choosing starting (source) node at random

RWP[s\_node] += 1

outlinked\_neigh = G.out\_edges(s\_node)

c = 0 # Iteration variable

while (c != 100000):

if (len(outlinked\_neigh) == 0): # Case when the node has no outlink

focus = random.choice(nodes) # focus is the node currently being visited

else:

random\_neigh = random.choice(list(outlinked\_neigh))

focus = random\_neigh[1]

RWP[focus] += 1

outlinked\_neigh = G.out\_edges(focus)

c += 1

return RWP

# Function for main program

def main():

G = nx.DiGraph() # Creating a directed graph object

N = 10 # Number of nodes in the graph

p = 0.3 # Probability for generating random graph

G.add\_nodes\_from([i for i in range(N)]) # Adding nodes to the graph object

G = add\_edges(G,p) # Adding edges to the graph object, making it a random graph

RWP\_list = random\_walk(G) # Performing random walks on the graph and getting list of RWP for nodes

nodes\_sorted\_by\_RWP = get\_nodes\_sorted\_by\_RWP(RWP\_list) # Ranking (ordering) nodes on the basis of accumulated RWP

print 'Nodes sorted (ranked) according to the accumulated random-walk-points : ',

for i in nodes\_sorted\_by\_RWP:

print i,

pr = nx.pagerank(G) # Getting PageRank of nodes, output is a dictionary

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

print '\nNodes sorted (ranked) using the builtin PageRank algo function : ',

for i in pr\_sorted:

print i[0],

main()