**EXPERIMENT NO. 10**

**DWM**

*# Implementation of Page Rank Algorithm # Tools used : Anaconda 3 jupyter* import numpy as np

from scipy.sparse import csc\_matrix

**def** pageRank(G, s = .85, maxerr = .0001):

*"""*

*Computes the pagerank for each of the n states Parameters*

*G: matrix representing state transitions*

*Gij is a binary value representing a transition from state i to j. s: probability of following a transition. 1-s probability of*

*teleporting*

*to another state.*

*maxerr: if the sum of pageranks between iterations is bellow this we will*

*have converged. """*

n = G.shape[0]

*# transform G into markov matrix A* A = csc\_matrix(G,dtype=np.float) rsums = np.array(A.sum(1))[:,0] ri, ci = A.nonzero()

A.data /= rsums[ri]

*# bool array of sink states*

sink = rsums==0

*# Compute pagerank r until we converge*

ro, r = np.zeros(n), np.ones(n)

**while** np.sum(np.abs(r-ro)) > maxerr: ro = r.copy()

*# calculate each pagerank at a time*

**for** i **in** range(0,n):

*# inlinks of state i*

Ai = np.array(A[:,i].todense())[:,0]

*# account for sink states*

Di = sink / float(n)

*# account for teleportation to state i*

Ei = np.ones(n) / float(n)

r[i] = ro.dot( Ai\*s + Di\*s + Ei\*(1-s) )

*# return normalized pagerank*

**return** r/float(sum(r))

**if** name ==' main ':

*# Example extracted from 'Introduction to Information Retrieval'*

G = np.array([[0,0,1,0,0,0,0],

[0,1,1,0,0,0,0],

[1,0,1,1,0,0,0],

[0,0,0,1,1,0,0],

[0,0,0,0,0,0,1],

[0,0,0,0,0,1,1],

[0,0,0,1,1,0,1]])

print(pageRank(G,s=.86))

[0.14285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714

0.14285714]

*# Implementation of HITS algorithm. # Implementation of HITS algorithm #Tools used : Anaconda 3*

*# importing modules*

import networkx as nx

import matplotlib.pyplot as plt G = nx.DiGraph()

G.add\_edges\_from([('A', 'D'), ('B', 'C'), ('B', 'E'), ('C', 'A'),

('D', 'C'), ('E', 'D'), ('E', 'B'), ('E', 'F'),

('E', 'C'), ('F', 'C'), ('F', 'H'), ('G', 'A'), ('G', 'C'), ('H', 'A')])

plt.figure(figsize =(10, 10)) nx.draw\_networkx(G, with\_labels = True)

hubs, authorities = nx.hits(G, max\_iter = 50, normalized = True)

*# The in-built hits function returns two dictionaries keyed by nodes # containing hub scores and authority scores respectively.* print("Hub Scores: ", hubs)

print("Authority Scores: ", authorities)

Hub Scores: {'A': 0.04642540386472174, 'D': 0.133660375232863, 'B': 0.15763599440595596, 'C': 0.037389132480584515, 'E':

0.2588144594158868, 'F': 0.15763599440595596, 'H':

0.037389132480584515, 'G': 0.17104950771344754} Authority Scores: {'A': 0.10864044085687284, 'D': 0.13489685393050574, 'B': 0.11437974045401585, 'C':

0.3883728005172019, 'E': 0.06966521189369385, 'F':

0.11437974045401585, 'H': 0.06966521189369385, 'G': 0.0}

