**ARYAMAN MISHRA**

**19BCE1027**

**Aim:**Implement HITS Algorithm.

**Data Structures Used:**2D Arrays.

**Algorithm:**

First, we have to explain what is authority and hub. HITS uses hubs and authorities to define a recursive relationship between webpages.

* Authority: A node is high-quality if many high-quality nodes link to it
* Hub: A node is high-quality if it links to many high-quality nodes

Algorithm Steps

* Initialize the hub and authority of each node with a value of 1
* For each iteration, update the hub and authority of every node in the graph
* The new authority is the **sum of the hub** of its parents
* The new hub is the **sum of the authority** of its children
* Normalize the new authority and hub

  Function to calculate the authority and hub score of all the nodes in the

  network.

  Parameters:

  outlinks: (n, n) int matrix where 1 represents the presence of a link and 0

    represents absence of a link

  Returns:  hub\_score: nd-array, containing the hub scores of the nodes

  authority\_score: nd-array, containing the authority scores of the nodes

1. Write a python program that takes the outlink data of a subsection of the web and that computes the Normalized Hub and authority score. The program should take the following parameters :
2. Number of sites in the sub-section of the web
3. Outlink data of the sites

**IMPLEMENTATION AND RESULTS:**

# Implementing HITS Algorithm

import numpy as np

# Function to calculate the authority and hub score of the nodes

def authority\_hub\_score(outlinks):

  # size of the matrix

  size = outlinks.shape[0]

  # Initializing the lists

  hub\_scores = [1.0 for i in range(size)]

  authority\_scores = [1.0 for i in range(size)]

  # Printing initial Hub scores

  print(hub\_scores)

  for \_ in range(100):

    # Calculating the authority scores of the nodes

    for j in range(size):

      temp\_auth = 0.0

      for i in range(size):

        if outlinks[i][j] == 1:

          temp\_auth += hub\_scores[i]

      authority\_scores[j] = temp\_auth

    # Normalizing the authority scores

    auth\_sum = sum(authority\_scores)

    # print(auth\_sum)

    for i in range(len(authority\_scores)):

      authority\_scores[i] /= auth\_sum

    # Calculating the hub scores of the nodes

    for i in range(size):

      temp\_hub = 0.0

      for j in range(size):

        if outlinks[i][j] == 1:

          temp\_hub += authority\_scores[j]

      hub\_scores[i] = temp\_hub

    # Normalizing the hub scores

    hub\_sum = sum(hub\_scores)

    # print(hub\_sum)

    for i in range(len(hub\_scores)):

      hub\_scores[i] /= hub\_sum

  return authority\_scores, hub\_scores

n = int(input('Enter the size of the matrix:\t'))

outlinks = []

for i in range(n\*n):

  temp = int(input('Enter the element:\t'))

  outlinks.append(temp)

outlinks = np.reshape(outlinks, (n, n))

authority\_scores, hub\_scores = authority\_hub\_score(outlinks)

print("Authority Scores:")

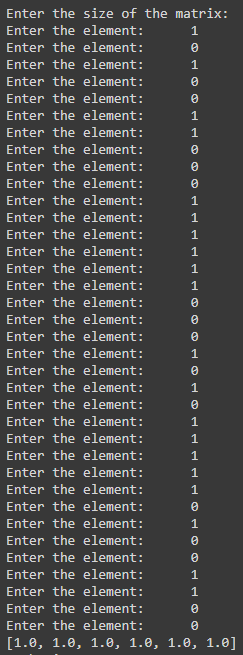
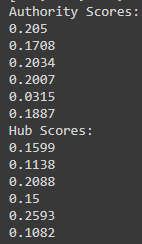
for i in (authority\_scores):

  print(round(i, 4))

print("Hub Scores:")

for i in (hub\_scores):

  print(round(i, 4))

1. Test your program using the following linkage data. Assume each site consists of only one text page.
2. Site A(outlinks to B,C,D)
3. Site B(outlinks to A,C,D)
4. Site C(outlinks to A,D)
5. Site D(outlinks to C,D,E)
6. Site E(outlinks to B,C,D)
7. Site F(outlinks to A)

**IMPLEMENTATION AND RESULTS:**

# Implementing HITS Algorithm

import numpy as np

# Function to calculate the authority and hub score of the nodes

def authority\_hub\_score(outlinks):

  # size of the matrix

  size = outlinks.shape[0]

  # Initializing the lists

  hub\_scores = [1.0 for i in range(size)]

  authority\_scores = [1.0 for i in range(size)]

  # Printing initial Hub scores

  print(hub\_scores)

  for \_ in range(100):

    # Calculating the authority scores of the nodes

    for j in range(size):

      temp\_auth = 0.0

      for i in range(size):

        if outlinks[i][j] == 1:

          temp\_auth += hub\_scores[i]

      authority\_scores[j] = temp\_auth

    # Normalizing the authority scores

    auth\_sum = sum(authority\_scores)

    # print(auth\_sum)

    for i in range(len(authority\_scores)):

      authority\_scores[i] /= auth\_sum

    # Calculating the hub scores of the nodes

    for i in range(size):

      temp\_hub = 0.0

      for j in range(size):

        if outlinks[i][j] == 1:

          temp\_hub += authority\_scores[j]

      hub\_scores[i] = temp\_hub

    # Normalizing the hub scores

    hub\_sum = sum(hub\_scores)

    # print(hub\_sum)

    for i in range(len(hub\_scores)):

      hub\_scores[i] /= hub\_sum

  return authority\_scores, hub\_scores

outlinks = [[0,1,1,1,0,0],[1,0,1,1,0,0],[1,0,0,1,0,0],[0,0,1,1,1,0],[0,1,1,1,0,0],[1,0,0,0,0,0]]

outlinks = np.reshape(outlinks, (6, 6))

authority\_scores, hub\_scores = authority\_hub\_score(outlinks)

print("Authority Scores:")

links=['A','B','C','D','E','F']

j=0

for i in (authority\_scores):

  print(links[j],":",round(i, 4))

  j=j+1

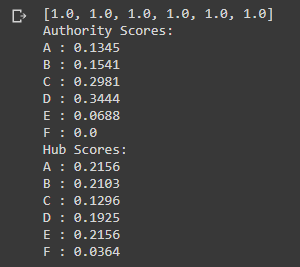
j=0

print("Hub Scores:")

for i in (hub\_scores):

  print(links[j],":",round(i, 4))

  j=j+1



CONCLUSION:HITS ALGORITHM HAS BEEN SUCCESFULLY IMPLEMENTED AND EXECUTED.