**Kulvir Singh**

**19BCE2074**

# Digital Assignment 3

## *Question 1 – Inverted Index*

### *Problem Statement :*

Write a python program to construct Inverted Index for any three English rhymes.

### *Procedure :*

Import nltk library as: from nltk.tokenize import word_tokenize

Open the files with the rhymes in them and store them into a variable string.

Remove punctuations from the rhymes.

Convert all the rhymes to lowercase.

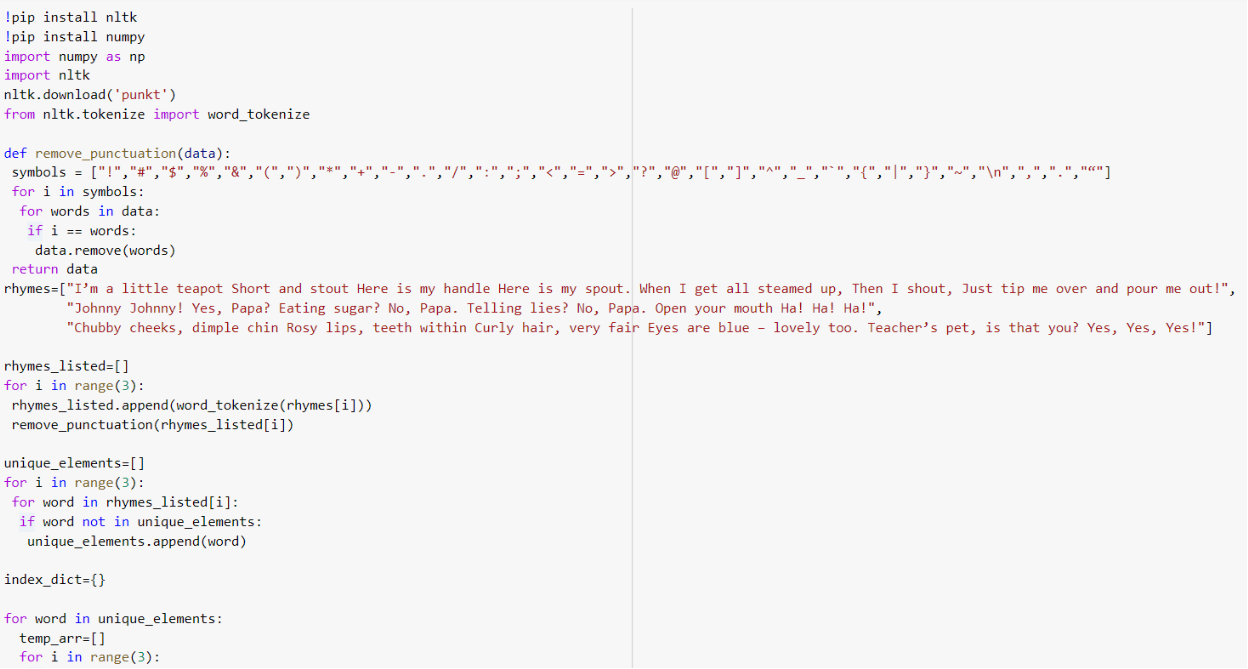
Write the logic for getting the desired output (as is mentioned in code section)

### *Code :*

```
!pip install nltk
!pip install numpy
import numpy as np
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
def remove_punctuation(data):
 symbols = ["!","#","$","%","&","(",")","*","+","-
",".","/",":",";","<","=",">","?","@","[","]","^","_","`","{","|","}","~",
"\n",",",".","“"]
 for i in symbols:
  for words in data:
   if i == words:
    data.remove(words)
 return data
rhymes=["I’m a little teapot Short and stout Here is my handle Here is my
spout. When I get all steamed up, Then I shout, Just tip me over and pour
me out!",
        "Johnny Johnny! Yes, Papa? Eating sugar? No, Papa. Telling lies? No, Papa. Open your mouth Ha! Ha! Ha!",
        "Chubby cheeks, dimple chin Rosy lips, teeth within Curly hair, very fair Eyes are blue - lovely too. Teacher’s pet, is that you? Yes, Yes, Yes!"]
rhymes_listed=[]
for i in range(3):
 rhymes_listed.append(word_tokenize(rhymes[i]))
 remove_punctuation(rhymes_listed[i])

unique_elements=[]
for i in range(3):
 for word in rhymes_listed[i]:
  if word not in unique_elements:
   unique_elements.append(word)

index_dict={}

for word in unique_elements:
  temp_arr=[]
  for i in range(3):
   if word in rhymes_listed[i]:
     temp_arr.append("id"+str(i+1))
  index_dict[word]=temp_arr
  print(word+": ",end='')
  for i in range (len(index_dict[word])):
   print(index_dict[word][i],end=" ")
  print("\n")
for word in unique_elements:
 key_position=[]
 word_doc_arr=[]
 for i in range (len(index_dict[word])):
  word_doc_arr.append(int(index_dict[word][i].split('id')[1])-1)
 print(word+": ",end='')
 index_dict_counter=0

 for j in word_doc_arr:
  key_position.clear()
  counter=1
  term_counter=0
  print("<",end="")
  print(index_dict[word][index_dict_counter],end=', ')
  index_dict_counter= index_dict_counter+1

  for words in rhymes_listed[j]:
```

```
  if word==words:
   term_counter=term_counter+1
 print(term_counter,end=', ')
 for words in rhymes_listed[j]:
  if word==words:
   key_position.append(counter)
  counter=counter+1
 print(key_position, end='')
 print(">",end="  ")
print("\n")
```

## *Code Screenshot:*



```
!pip install nltk
!pip install numpy
import numpy as np
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize

def remove_punctuation(data):
 symbols = ["!","#","$","%","&","(",")","*","+","-",".","/",":",";","<","=",">","?","@","[","]","^","_","`","{","|","}","~","\n",",",".","“"]
 for i in symbols:
  for words in data:
   if i == words:
    data.remove(words)
 return data
rhymes=["I’m a little teapot Short and stout Here is my handle Here is my spout. When I get all steamed up, Then I shout, Just tip me over and pour me out!",
        "Johnny Johnny! Yes, Papa? Eating sugar? No, Papa. Telling lies? No, Papa. Open your mouth Ha! Ha! Ha!",
        "Chubby cheeks, dimple chin Rosy lips, teeth within Curly hair, very fair Eyes are blue – lovely too. Teacher’s pet, is that you? Yes, Yes, Yes!"]

rhymes_listed=[]
for i in range(3):
 rhymes_listed.append(word_tokenize(rhymes[i]))
 remove_punctuation(rhymes_listed[i])

unique_elements=[]
for i in range(3):
 for word in rhymes_listed[i]:
  if word not in unique_elements:
   unique_elements.append(word)

index_dict={}

for word in unique_elements:
  temp_arr=[]
  for i in range(3):
   if word in rhymes_listed[i]:
     temp_arr.append("id"+str(i+1))
  index_dict[word]=temp_arr
  print(word+": ",end='')
  for i in range (len(index_dict[word])):
   print(index_dict[word][i],end=" ")
  print("\n")
for word in unique_elements:
 key_position=[]
 word_doc_arr=[]
 for i in range (len(index_dict[word])):
  word_doc_arr.append(int(index_dict[word][i].split('id')[1])-1)
 print(word+": ",end='')
 index_dict_counter=0

 for j in word_doc_arr:
  key_position.clear()
  counter=1
  term_counter=0
  print("<",end="")
  print(index_dict[word][index_dict_counter],end=', ')
  index_dict_counter= index_dict_counter+1

  for words in rhymes_listed[j]:
   if word==words:
    term_counter=term_counter+1
  print(term_counter,end=', ')
  for words in rhymes_listed[j]:
   if word==words:
    key_position.append(counter)
   counter=counter+1
  print(key_position, end='')
  print(">",end=" ")
 print("\n")
```

### ***Output Screenshots :***

```
I: id1

’: id1 id3

m: id1

a: id1
```

```
little: id1
teapot: id1
Short: id1
and: id1
stout: id1
Here: id1
is: id1 id3
my: id1
handle: id1
spout: id1
When: id1
get: id1
all: id1
steamed: id1
up: id1
Then: id1
shout: id1
Just: id1
tip: id1
me: id1
over: id1
pour: id1
out: id1
Johnny: id2
Yes: id2 id3
Papa: id2
Eating: id2
```

```
sugar: id2

No: id2

Telling: id2

lies: id2

Open: id2

your: id2

mouth: id2

Ha: id2

Chubby: id3

cheeks: id3

dimple: id3

chin: id3

Rosy: id3

lips: id3

teeth: id3

within: id3

Curly: id3

hair: id3

very: id3

fair: id3

Eyes: id3

are: id3

blue: id3

-: id3

lovely: id3

too: id3

Teacher: id3
```

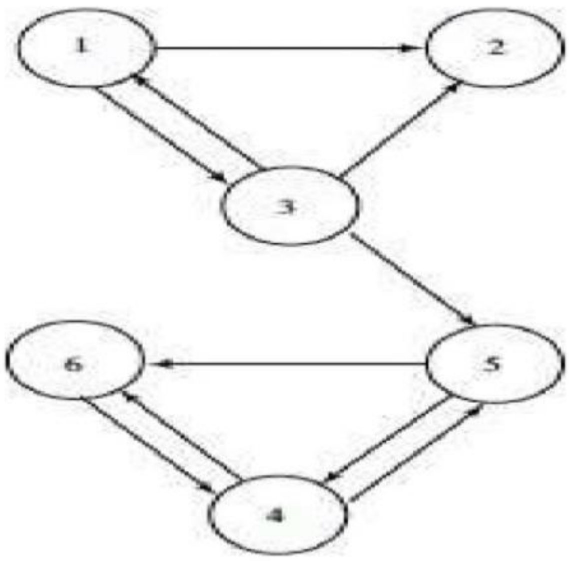
```
s: id3
pet: id3
that: id3
you: id3
I: <id1, 3, [1, 19, 25]>
’: <id1, 1, [2]> <id3, 1, [20]>
m: <id1, 1, [3]>
a: <id1, 1, [4]>
little: <id1, 1, [5]>
teapot: <id1, 1, [6]>
Short: <id1, 1, [7]>
and: <id1, 2, [8, 31]>
stout: <id1, 1, [9]>
Here: <id1, 2, [10, 14]>
is: <id1, 2, [11, 15]> <id3, 1, [23]>
my: <id1, 2, [12, 16]>
handle: <id1, 1, [13]>
spout: <id1, 1, [17]>
When: <id1, 1, [18]>
get: <id1, 1, [20]>
all: <id1, 1, [21]>
steamed: <id1, 1, [22]>
up: <id1, 1, [23]>
Then: <id1, 1, [24]>
shout: <id1, 1, [26]>
Just: <id1, 1, [27]>
tip: <id1, 1, [28]>
```

```
me: <id1, 2, [29, 33]>
over: <id1, 1, [30]>
pour: <id1, 1, [32]>
out: <id1, 1, [34]>
Johnny: <id2, 2, [1, 2]>
Yes: <id2, 1, [3]> <id3, 3, [26, 27, 28]>
Papa: <id2, 3, [4, 8, 12]>
Eating: <id2, 1, [5]>
sugar: <id2, 1, [6]>
No: <id2, 2, [7, 11]>
Telling: <id2, 1, [9]>
lies: <id2, 1, [10]>
Open: <id2, 1, [13]>
your: <id2, 1, [14]>
mouth: <id2, 1, [15]>
Ha: <id2, 3, [16, 17, 18]>
Chubby: <id3, 1, [1]>
cheeks: <id3, 1, [2]>
dimple: <id3, 1, [3]>
chin: <id3, 1, [4]>
Rosy: <id3, 1, [5]>
lips: <id3, 1, [6]>
teeth: <id3, 1, [7]>
within: <id3, 1, [8]>
Curly: <id3, 1, [9]>
hair: <id3, 1, [10]>
very: <id3, 1, [11]>
```

```
fair: <id3, 1, [12]>

Eyes: <id3, 1, [13]>

are: <id3, 1, [14]>

blue: <id3, 1, [15]>

-: <id3, 1, [16]>

lovely: <id3, 1, [17]>

too: <id3, 1, [18]>

Teacher: <id3, 1, [19]>

s: <id3, 1, [21]>

pet: <id3, 1, [22]>

that: <id3, 1, [24]>

you: <id3, 1, [25]>
```

# *Question 2 – Page Rank*

## *Problem Statement :*



Find the Page Rank of the given graph up to 7 iterations.

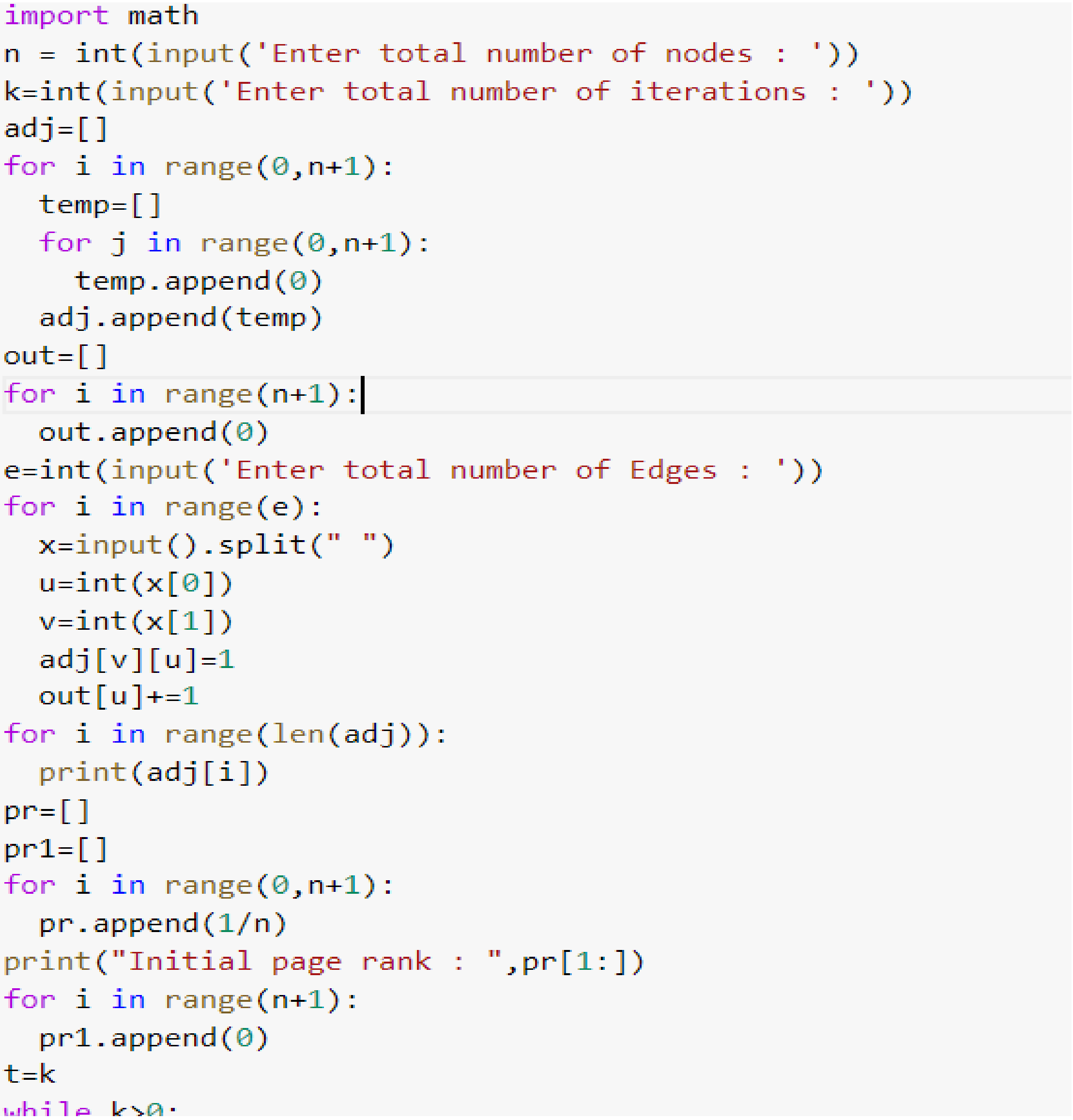
## *Procedure :*

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

## *Code :*

```
import math
n = int(input('Enter total number of nodes : '))
k=int(input('Enter total number of iterations : '))
adj=[]
for i in range(0,n+1):
  temp=[]
  for j in range(0,n+1):
    temp.append(0)
  adj.append(temp)
out=[]
for i in range(n+1):
  out.append(0)
e=int(input('Enter total number of Edges : '))
for i in range(e):
  x=input().split(" ")
  u=int(x[0])
  v=int(x[1])
  adj[v][u]=1
  out[u]+=1
for i in range(len(adj)):
  print(adj[i])
pr=[]
pr1=[]
for i in range(0,n+1):
  pr.append(1/n)
print("Initial page rank : ",pr[1:])
for i in range(n+1):
  pr1.append(0)
t=k
while k>0:
  for i in range(n+1):
    pr1[i]=0
  for i in range(1,n+1):
    for j in range(len(adj[i])):
      if adj[i][j]==1:
        pr1[i]+= (pr[j]/out[j])
  for i in range(n+1):
    pr[i]=pr1[i]
  print("Iteration", t-k+1 ,"page rank : ",pr[1:])
  k-=1
m=0
highest_pr=0
for i in range(n+1):
  if pr[i]>m:
    m=pr[i]
    high_pr=i
print("\nHighest Page Rank is ", high_pr," with page rank: ",m)
```

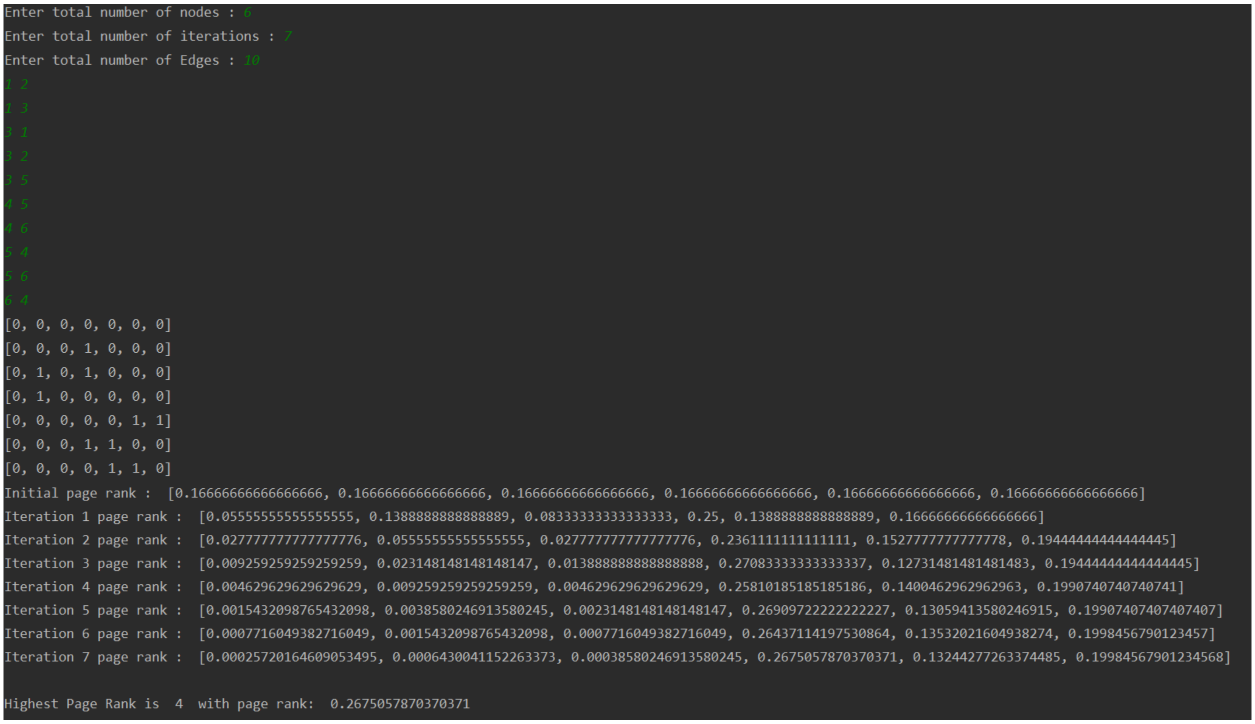
***Code Screenshot:***



```
import math
n = int(input('Enter total number of nodes : '))
k=int(input('Enter total number of iterations : '))
adj=[]
for i in range(0,n+1):
  temp=[]
  for j in range(0,n+1):
    temp.append(0)
  adj.append(temp)
out=[]
for i in range(n+1):
  out.append(0)
e=int(input('Enter total number of Edges : '))
for i in range(e):
  x=input().split(" ")
  u=int(x[0])
  v=int(x[1])
  adj[v][u]=1
  out[u]+=1
for i in range(len(adj)):
  print(adj[i])
pr=[]
pr1=[]
for i in range(0,n+1):
  pr.append(1/n)
print("Initial page rank : ",pr[1:])
for i in range(n+1):
  pr1.append(0)
t=k
while k>0:
```

```
while k>0:
  for i in range(n+1):
    pr1[i]=0
  for i in range(1,n+1):
    for j in range(len(adj[i])):
      if adj[i][j]==1:
        pr1[i]+= (pr[j]/out[j])
  for i in range(n+1):
    pr[i]=pr1[i]
  print("Iteration", t-k+1 ,"page rank : ",pr[1:])
  k-=1
m=0
highest_pr=0
for i in range(n+1):
  if pr[i]>m:
    m=pr[i]
    high_pr=i
print("\nHighest Page Rank is ", high_pr," with page rank: ",m)
```

## *Output Screenshots :*



```
Enter total number of nodes : 6
Enter total number of iterations : 7
Enter total number of Edges : 10
1 2
1 3
3 1
3 2
3 5
4 5
4 6
5 4
5 6
6 4
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0]
[0, 1, 0, 1, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 1]
[0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 1, 1, 0]
Initial page rank :  [0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
Iteration 1 page rank :  [0.05555555555555555, 0.1388888888888889, 0.08333333333333333, 0.25, 0.1388888888888889, 0.16666666666666666]
Iteration 2 page rank :  [0.027777777777777776, 0.05555555555555555, 0.027777777777777776, 0.2361111111111111, 0.1527777777777778, 0.19444444444444445]
Iteration 3 page rank :  [0.009259259259259259, 0.023148148148148147, 0.013888888888888888, 0.27083333333333337, 0.12731481481481483, 0.19444444444444445]
Iteration 4 page rank :  [0.004629629629629629, 0.009259259259259259, 0.004629629629629629, 0.25810185185185186, 0.1400462962962963, 0.19907407407407407]
Iteration 5 page rank :  [0.0015432098765432098, 0.0038580246913580245, 0.0023148148148148147, 0.26909722222222227, 0.13059413580246915, 0.19907407407407407]
Iteration 6 page rank :  [0.0007716049382716049, 0.0015432098765432098, 0.0007716049382716049, 0.26437114197530864, 0.13532021604938274, 0.1998456790123457]
Iteration 7 page rank :  [0.00025720164609053495, 0.0006430041152263373, 0.00038580246913580245, 0.2675057870370371, 0.13244277263374485, 0.19984567901234568]

Highest Page Rank is  4  with page rank:  0.2675057870370371
```

# *Question 3 – HITS*

## ***Problem Statement :***

Implement HITS Algorithm for the given graph in question 1 for 7 iterations.

## ***Procedure :***

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

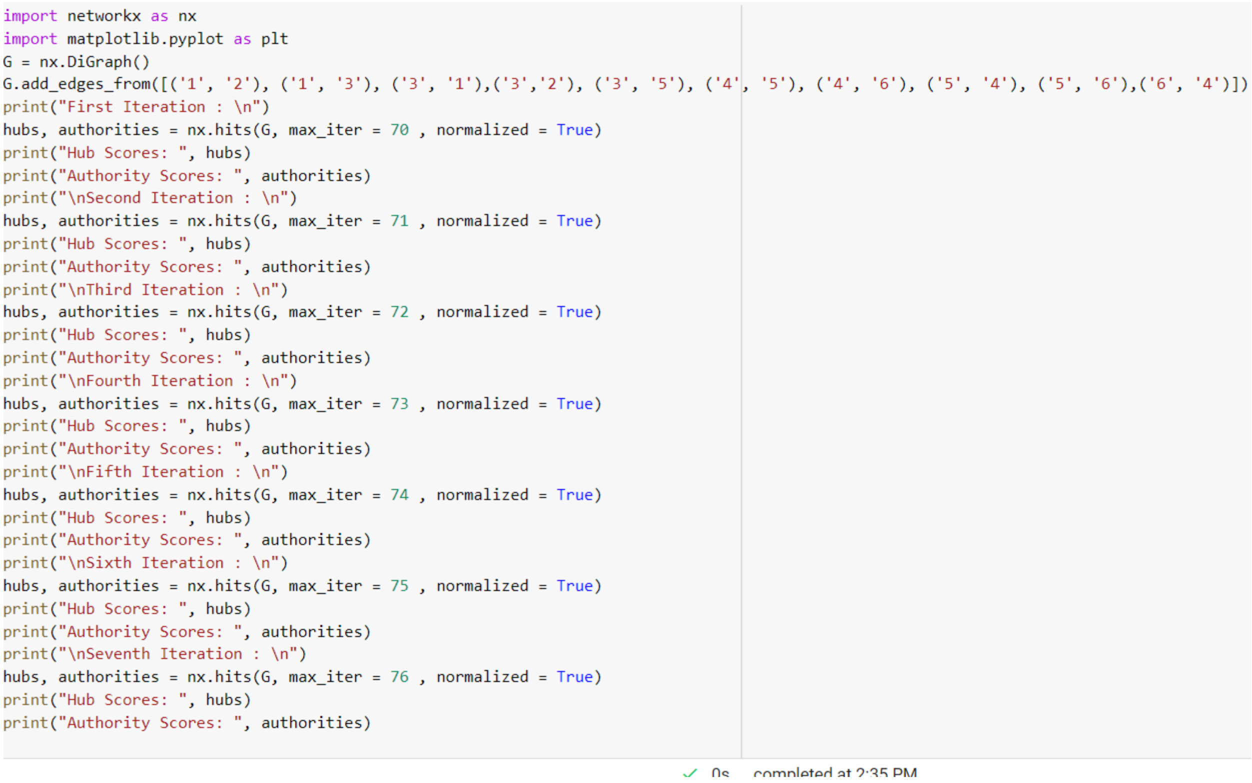
Normalize the new authority and hub

## ***Code :***

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()

G.add_edges_from([('1', '2'), ('1', '3'), ('3', '1'),('3', '5'), ('4', '5'
), ('4', '6'), ('5', '4'), ('5', '6'),('6', '4')])


print("First Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 70 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nSecond Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 71 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nThird Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 72 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nFourth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 73 , normalized = True)
```

```
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nFifth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 74 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nSixth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 75 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)

print("\nSeventh Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 76 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
```

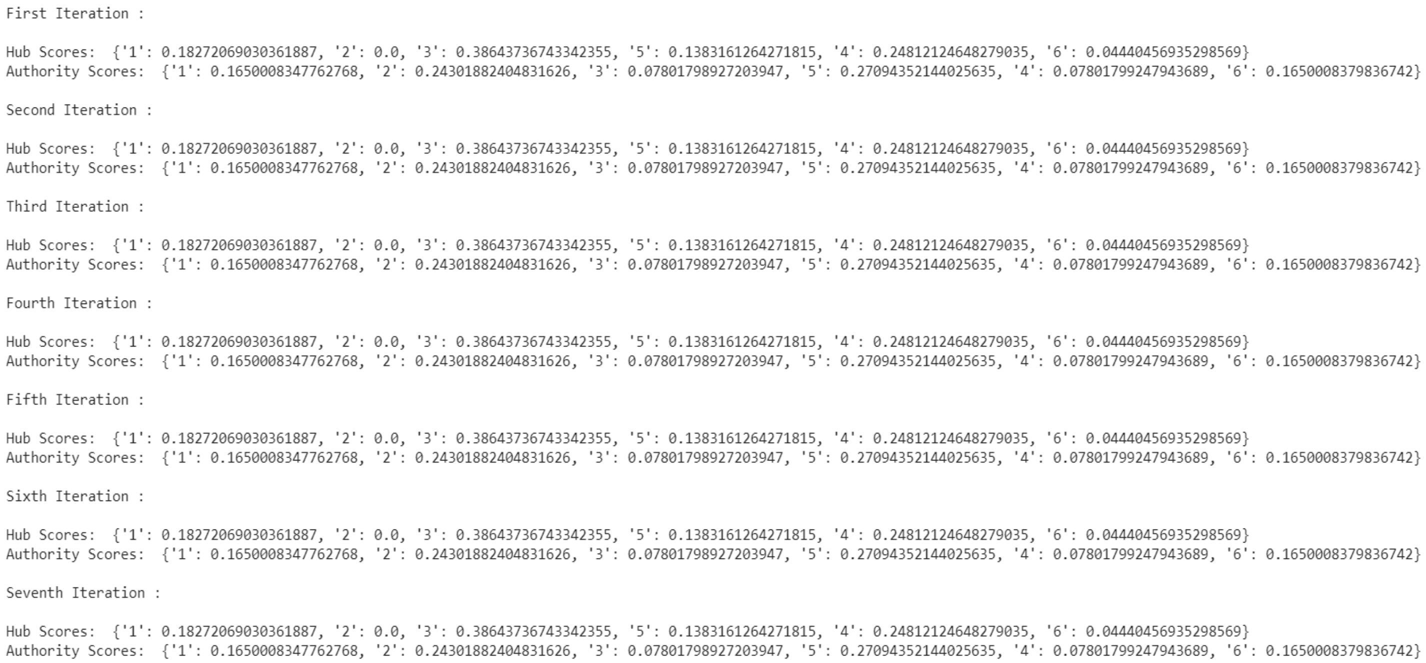
## *Code Screenshot:*



```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_edges_from([('1', '2'), ('1', '3'), ('3', '1'),('3','2'), ('3', '5'), ('4', '5'), ('4', '6'), ('5', '4'), ('5', '6'),('6', '4')])
print("First Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 70 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nSecond Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 71 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nThird Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 72 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nFourth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 73 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nFifth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 74 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nSixth Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 75 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
print("\nSeventh Iteration : \n")
hubs, authorities = nx.hits(G, max_iter = 76 , normalized = True)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
```

✓ 0s completed at 2:35 PM

## *Output Screenshots :*



```
First Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Second Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Third Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Fourth Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Fifth Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Sixth Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}

Seventh Iteration :

Hub Scores:  {'1': 0.18272069030361887, '2': 0.0, '3': 0.38643736743342355, '5': 0.1383161264271815, '4': 0.24812124648279035, '6': 0.04440456935298569}
Authority Scores:  {'1': 0.1650008347762768, '2': 0.24301882404831626, '3': 0.07801798927203947, '5': 0.27094352144025635, '4': 0.07801799247943689, '6': 0.1650008379836742}
```